Software Implementation of Optimal Pairings on Elliptic Curves with Odd Prime Embedding Degrees

Yu Dai[1], Zijian Zhou[4,5], Fangguo Zhang[2,3], and ✉Chang-An Zhao[1,3]

[1] School of Mathematics, Sun Yat-sen University, Guangzhou 510275, P.R.China.{daiy39}@mail2.sysu.edu.cn
[2] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, P.R. China
[3] Guangdong Key Laboratory of Information Security, Guangzhou 510006, P.R. China. {zhaochan3}@mail.sysu.edu.cn
[4] College of Liberal Arts and Sciences National University of Defense Technology 410073 Changsha P.R.China
[5] Hunan Engineering Research Center of Commercial Cryptography Theory and Technology Innovation 410073 Changsha P.R.China

Abstract. Pairing computations on elliptic curves with odd prime degrees are rarely studied as low efficiency. Recently, Clarisse, Duquesne and Sanders proposed two new curves with odd prime embedding degrees: *BW13-P310* and *BW19-P286*, which are suitable for some special cryptographic schemes. In this paper, we propose efficient methods to compute the optimal ate pairing on this types of curves, instantiated by the *BW13-P310* curve. We first extend the technique of lazy reduction into the finite field arithmetic. Then, we present a new method to execute Miller's algorithm. Compared with the standard Miller iteration formulas, the new ones provide a more efficient software implementation of pairing computations. At last, we also give a fast formula to perform the final exponentiation. Our implementation results indicate that it can be computed efficiently, while it is slower than that over the BLS-446 curve at the same security level.

Keywords: Pairing Computations, Odd Prime Embedding Degree, Miller Iteration

1 Introduction

Pairings on elliptic curves are a powerful tool in cryptography because of their widespread applications in cryptographic schemes such as Identity-Based Encryption (IBE) [1], Short Signatures [2], Direct Anonymous Attestation (DAA) [3], and Enhanced Privacy ID (EPID) [4]. The book [5] provides a good reference of pairing-based cryptography. A pairing is a bilinear and non-degenerated map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two subgroups of order r of an elliptic curve, and $\mathbb{G}_T$ is a subgroup of order r of $\mathbb{F}_{p^k}$, where k is the embedding degree of the elliptic curve with respect to r. In order to shorten the length of Miller loop, several variants of Tate pairing are proposed, such as ate [6, 7], ate$_i$ [8], R-ate [9, 10] and optimal [11] pairings. For efficiency, most implementations of pairing computations are over elliptic curves with

embedding degrees $k = 2^i 3^j$. However, the special structure of $\mathbb{F}_{p^k}$ also results in some potential security risks. The security of pairing-based protocols is based on the hardness of discrete logarithm problems in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{F}_{p^k}$, respectively. Due to the Pollard's rho [12] algorithm, the security level in $\mathbb{G}_1$ and $\mathbb{G}_2$ is $\log_2 |r|/2$. To compute the discrete logarithm in $\mathbb{F}_{p^k}$ with p medium or large [13], the state-of-the-art attacks are variants of NFS. Gordon [14] and Schirokauer [15] first used NFS to attack the DLP in prime fields. Schirokauer [16] adapted the NFS and proposed a new variant, later known as TNFS, to compute the discrete logarithm in $\mathbb{F}_{p^k}$ with $k > 1$ based on a tower extension. This can be applied to determine the key sizes of pairing based cryptosystems. Joux *et al*. [17] later applied the NFS (known as JLSV) to finite fields for any large or medium characteristic p. In 2016, Kim and Barbulescu [13] introduced exTNFS to DLP in $\mathbb{F}_{p^k}$ with k a composite number mainly by modifying the polynomial selection in TNFS. This implies that for a non-prime number k, NFS has better complexity for p medium size. Furthermore, in paring-based construction, the prime p always has a special form of $P(u)/v$ with some polynomial $P(x) \in \mathbb{Z}[x]$ and some integers u, v. Kim and Barbulescu [13] proposed a variant of exTNFS, named SexTNFS, to target DLP in $\mathbb{F}_{p^k}$ when p has the special form, which greatly reduces the asymptotic complexity even compared to exTNFS. This implies that pairing-based cryptosystems associated with a composite number k require larger key sizes compared to those ones with k prime.

Hence, the curves with prime embedding degree may be preferred for the special schemes, such as EPID and DAA, which require an amount of exponentiations in $\mathbb{G}_1$. Clarisse *et al*. [18] studied pairing friendly curves with fast exponentiation in $\mathbb{G}_1$ and recommended two curves: *BW13-P310* and *BW19-P286*. To our knowledge, the software implementation of pairings on the two curves have never been studied. In this paper, we investigate efficient pairing computation on the curves with odd prime embedding degrees. An instantiation using *BW13-P310* curve is discussed in detail. We summarize our contributions as follows:

- We first investigate the finite field arithmetic in $\mathbb{F}_{p^{13}}$. The technique of lazy reduction is extended to the multiplication, squaring and inversion in $\mathbb{F}_{p^{13}}$. An efficient inversion operation is also presented.
- We introduce a modified Miller function, which is suitable for pairing computations with odd prime embedding degrees. On this basis, new iterative formulas are given to speed up the computation in the Miller loop.
- We examine the efficiency of pairing computation in Jacobian and homogeneous coordinates. The results show that the former one is more efficient than the latter.
- Finally, we optimize the computation of the final exponentiation. We implement the optimal ate pairing over the *BW13-P310* curve on a 64-bit PC platform. Our implementation results show that it can be computed efficiently, although it is not as efficient as that over the *BLS*-446 curve.

Outline of this paper: Section 2 gives a brief overview of the optimal ate pairing and the explicit formula of that over the *BW13-P310* curve. In Section 3, we describe the finite field arithmetic optimized by the technique of lazy reduction. Sections 4 and 5 study the modified Miller double-and-add and the modified Miller quadruple-and-add, respectively. In Section 6, we introduce an efficient method to implement the final exponentiation of the *BW13-P310* curve. Section 7 summarizes operation counts and

implementation results of the optimal ate pairing computation on the curve. We draw conclusions in Section 8.

2 Preliminaries

2.1 Optimal ate pairing

Let $\mathbb{F}_p$ be a prime field of characteristic p and E an elliptic curve defined over $\mathbb{F}_p$. Denote by O the infinity point of E. We use $\#E(\mathbb{F}_p)$ to denote the order of $E(\mathbb{F}_p)$. Then $\#E(\mathbb{F}_p) = p + 1 - t$, where t is the trace of the Frobenius endomorphism $\pi_p : (x, y) \rightarrow (x^p, y^p)$. Consider a large prime r such that $r|\#E(\mathbb{F}_p)$. Then the embedding degree k is the smallest positive integer such that $r|p^k - 1$. The number k also ensures $E[r] \subseteq E(\mathbb{F}_{p^k})$. Let $\mathbb{G}_1 = \{P \in E[r]|\pi_p(P) = P\}$ and $\mathbb{G}_2 = \{P \in E[r]|\pi_p(P) = [p]P\}$. Let m be an integer such that $r \nmid m$ and the coefficients of $\lambda = mr$ in basis p are as small as possible. Define the Miller function $f_{u,Q}$ to be the normalized rational function with divisor

$$div(f_{u,Q}) = u(Q) - ([u]Q) - (u - 1)(O),$$

where Q is a rational point of E. Write $\lambda = \sum_{i=0}^{l} c_i p^i$ and $s_i = \sum_{j=i}^{l} c_j p^j$. The general expression of the optimal ate pairing is defined as [11]:

$$a_{opt} : \mathbb{G}_2 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T,$$

$$(Q, P) \rightarrow \left(\prod_{i=0}^{l} f_{c_i,Q}^{p^i}(P) \cdot \prod_{i=0}^{l-1} \frac{\ell_{[s_{i+1}]Q,[c_i p^i]Q}(P)}{\nu_{[s_i]Q}(P)} \right)^{\frac{(p^k-1)}{r}},$$

where $\ell_{[s_{i+1}]Q,[c_i p^i]Q}$ is the straight line passing through the points $[s_{i+1}]Q$ and $[c_i p^i]Q$, and $\nu_{[s_i]Q}$ is the vertical line passing through the points $[s_i]Q$ and $[-s_i]Q$. The target coefficients c_i can be captured from the short vectors of the following lattice L.

$$L = \begin{pmatrix} r & 0 & 0 & \cdots & 0 \\ -p & 1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -p^{\varphi(k)-1} & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

2.2 *BW13-P310* curve

The improvement of NFS results in a larger field size for pairing friendly curves at 128-bit security level. New security estimation of pairing friendly curves are given in [19, 20]. If we restrict the embedding degree $k < 19$ and the computer word-size $w = 64$, the curve from Construction 6.6 in [21] with $k = 13$ is the sole survivor such that the prime p can be represented by 5 words. The order $r(u)$, the prime $p(u)$ and the Frobenius trace $t(u)$ of the curve are given by

$$\begin{aligned} r(u) &= \Phi_{78}(u), \\ p(u) &= \tfrac{1}{3}(u + 1)^2(u^{26} - u^{13} + 1) - u^{27}, \\ t(u) &= -u^{14} + u + 1. \end{aligned}$$

By using the function *ShortestVectors* in MAGMA [22], we obtain a short vector of lattice L as $C = [u^2, -u, 1, 0, 0, 0, 0, 0, 0]$. The seed u is selected as $u = -0x80b$ and the corresponding elliptic curve is given as

$$E : y^2 = x^3 - b, \tag{1}$$

where $b = 17$. As the curve is actually constructed by using the Brezing-Weng method with embedding degree $k = 13$, and the bit length of the prime p is 310, it is named as *BW13-P310* in [18]. In the following, we denote $x = -u = 0x80b$. Then the optimal pairing is expressed as

$$a_{opt} = (f_{x^2,Q} \cdot f_{x,Q}^{p} \cdot \ell_{\pi^2(Q),\pi([x]Q)})^{(p^{13}-1)/r}.$$

According to [5, Lemma 3.5], we find that

$$f_{x^2,Q} \cdot f_{x,Q}^{p} = f_{x,Q}^{x+p} \cdot f_{x,[x]Q}.$$

Therefore, the optimal pairing can be equivalently written as

$$a_{opt} = (f_{x,Q}^{x+p} \cdot f_{x,[x]Q} \cdot \ell_{\pi^2(Q),\pi([x]Q)})^{(p^{13}-1)/r},$$

which gives a short Miller loop.

2.3 Pairing computation

Pairing computation consists of the *Miller loop*, which mainly computes $f_{c_i,Q}(P)$ by using Miller's Algorithm (Algorithm 1), and the *final exponentiation*, which raises the result of the Miller loop to the power of $d = \frac{p^k-1}{r}$. Miller iteration is based on the following relations:

$$\begin{aligned} f_{1,Q} &= 1, \\ f_{u+1,Q} &= f_{u,Q} \cdot \frac{\ell_{[u]Q,Q}}{\nu_{[u+1]Q}}, \\ f_{2u,Q} &= f_{u,Q}^2 \cdot \frac{\ell_{[u]Q,[u]Q}}{\nu_{[2u]Q}}. \end{aligned}$$

When the embedding degree k is composite, the computation of the final exponentiation is usually relatively easy [23–25]. When k is prime, the exponent d could be factorized as $d = (p-1) \cdot \eta$, where $\eta = \frac{\Phi_k(p)}{r}$, and Φ_k denotes the k-th cyclotomic polynomial. Raising an element $f \in \mathbb{F}_{p^k}$ to the power $p-1$ is easy via one p-Frobenius map, one inversion and one multiplication in $\mathbb{F}_{p^k}$. However, raising f to the power η is relatively costly. Generally, the exponent η can be expressed as $\eta{=}\lambda_0 + \lambda_1 p \ldots \lambda_{k-1} p^{k-2}$. Note that a fixed non-degenerate power of a pairing is still a pairing. Therefore, we can equivalently compute $f^{\eta'}$, where $\eta' = m \cdot \eta$ for some m with $r \nmid m$ and the coefficients of η' in basis p are as small as possible. More details are shown in [26]. In Section 6, we will investigate the final exponentiation on the *BW13-P310* curve in detail.

Algorithm 1: MillerLoop(x, Q, P)

Input: $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, x = \sum_{i=0}^{\lfloor log_2 x \rfloor} x_i 2^i$
Output: $f_{x,Q}(P)$
$T \leftarrow Q, f \leftarrow 1$
for $i = \lfloor \log_2 x \rfloor - 1$ **downto** 0 **do**
  $f \leftarrow f^2 \cdot \frac{\ell_{T,T}(P)}{\nu_{2T}(P)}, T \longleftarrow 2T$
  if $x_i = 1$ **then**
    $f \leftarrow f \cdot \frac{\ell_{T,Q}(P)}{\nu_{T+Q}(P)}, T \leftarrow T + Q$
  end if
  if $x_i = -1$ **then**
    $f \leftarrow f \cdot \frac{\ell_{T,-Q}(P)}{\nu_{T-Q}(P)}, T \leftarrow T - Q$
  end if
end for
return f

3 Finite Field Operations in $\mathbb{F}_{p^{13}}$

For any embedding degree k, the extension field $\mathbb{F}_{p^k}$ can be seen as $\mathbb{F}_p[\omega]/\langle f(\omega)\rangle$, where $f(\omega)$ is a k-th irreducible polynomial over $\mathbb{F}_p$. The irreducible polynomial $f(\omega)$ is usually chosen as a binomial in order to implement finite field arithmetic efficiently. From [27, Theorem 3.75], $f(\omega)$ can be chosen as a binomial if $p \equiv 1 \bmod k$. Fortunately, the prime selected for the *BW13-P310* curve meets this requirement. Hence, the extension field $\mathbb{F}_{p^{13}}$ can be constructed as $\mathbb{F}_{p^{13}} = \mathbb{F}_p[\omega]/\langle \omega^{13} - \xi\rangle$ for some $\xi \in \mathbb{F}_p$. For the prime p selected, the parameter ξ can be chosen as $\xi = 2$. In the section, we mainly discuss the finite field arithmetic operations in $\mathbb{F}_{p^{13}}$, including multiplication, squaring, inversion and Frobenius operations. We skip the discussion of the arithmetic of addition/subtraction in $\mathbb{F}_{p^{13}}$ since it is relatively simple and straightforward. All notations used in the following field operations are presented in Table 1.

3.1 Lazy reduction

Lazy reduction technique was first introduced in pairing computations by [28] to speed up the multiplication in $\mathbb{F}_{p^2}$. Aranha *et al*. [23] extended the method to the whole pairing computations in tower-friendly fields. By using the technique the computation of $a \times b \bmod p \oplus c \times d \bmod p$ is replaced by that of $(a \times b + c \times d) \bmod p$ for any $a, b \in \mathbb{F}_p$. Hence one modular reduction can be saved. It should be noted that $a \times b \bmod p$ and $c \times d \bmod p$ are single-precision numbers (occupying 5 words for the selected p), and $a \times b$ and $c \times d$ are double-precision numbers (occupying 10 words for the selected p). The shortcoming of the technique is that one single-precision addition is replaced by one double-precision addition. Let $N = w \cdot \left\lceil \frac{\lceil \log_2 p \rceil}{w} \right\rceil$. As the upper bound of Montgomery reduction is $2^N \cdot p$, it should be careful when using the technique of lazy reduction. In this section, we apply the technique to the arithmetic in $\mathbb{F}_{p^{13}}$.

3.2 Multiplication

For any $a = \sum_{i=0}^{12} a_i\omega^i$ and $b = \sum_{i=0}^{12} b_i\omega^i$, the computation of $a \cdot b$ is given in Algorithm 2 by using Karatsuba multiplication [29]. The corresponding sub-algorithms of *MulLevel-i*(a, b) for $i \in \{2, 3, 4, 6, 7\}$ are used to compute $\sum_{j=0}^{i} a_j\omega^j \times \sum_{j=0}^{i} b_j\omega^j$, which are presented in *Appendix A*. Now we explain every step of the algorithm. At Lines 1-4, the two elements a and b are written as $a = A_0 + A_1\omega^6$ and $b = B_0 + B_1\omega^6$. Then, we have

$$a \times b = \alpha + (\gamma - \alpha - \beta)\omega^6 + \beta\omega^{12},$$

where α, β and γ are given at Lines $5-9$. Lines $10-15$ compute $\gamma - \alpha - \beta$. It is easy to check that $\gamma_i - (\alpha_i + \beta_i) > 0$ for $i \in \{0, \ldots, 12\}$. Thus, the above double precision subtractions do not require carry checks. Lines $16-26$ compute $a \times b$. As ξ is small, the values of $u_i \in (0, 2^N \cdot p)$ for $i \in \{0, \ldots, 12\}$. Hence, the above double precision additions also do not require carry checks. Lines $27-29$ compute $a \cdot b = a \times b \bmod p$. The algorithm of squaring in $\mathbb{F}_{p^{13}}$ is similar. Here we do not discuss it in detail as it is tedious. The operation counts of multiplication and squaring in $\mathbb{F}_{p^{13}}$ are summarized in Table 4.

Algorithm 2: Multiplication in $\mathbb{F}_{p^{13}}$

1 **Input:** $a = \sum_{i=0}^{12} a_i\omega^i, b = \sum_{i=0}^{12} b_i\omega^i$,

2 **Output:** $c = a \cdot b = \sum_{i=0}^{12} c_i\omega^i \in \mathbb{F}_{p^{13}}$

1: $A_0 \leftarrow \sum_{i=0}^{5} a_i\omega^i$
2: $A_1 \leftarrow \sum_{i=0}^{6} a_{i+6}\omega^i$
3: $B_0 \leftarrow \sum_{i=0}^{5} b_i\omega^i$
4: $B_1 \leftarrow \sum_{i=0}^{6} b_{i+6}\omega^i$
5: $\alpha \leftarrow MulLevel6(A_0, B_0)$
6: $\beta \leftarrow MulLevel7(A_1, B_1)$
7: $T_0 \leftarrow A_0 + A_1$
8: $T_1 \leftarrow B_0 + B_1$
9: $\gamma \leftarrow MulLevel7(T_0, T_1)$
10: **for** $i = 0$ **to** 10 **do**
11: $\quad \gamma_i \leftarrow \gamma_i - \alpha_i$
12: $\quad \gamma_i \leftarrow \gamma_i - \beta_i$
13: **end for**
14: $\gamma_{11} \leftarrow \gamma_{11} - \beta_{11}$
15: $\gamma_{12} \leftarrow \gamma_{12} - \beta_{12}$
16: **for** $i = 0$ **to** 5 **do**
17: $\quad u_i \leftarrow \beta_{i+1} + \gamma_{i+7}$
18: $\quad u_i \leftarrow \xi \cdot u_i$
19: $\quad u_i \leftarrow \alpha_i + u_i$
20: **end for**
21: **for** $i = 6$ **to** 10 **do**
22: $\quad u_i \leftarrow \alpha_i + \gamma_{i-6}$
23: $\quad u_i \leftarrow u_i + \xi \cdot \beta_{i+1}$
24: **end for**
25: $u_{11} \leftarrow \gamma_5 + \xi \cdot \beta_{12}$
26: $u_{12} \leftarrow \gamma_6 + \beta_0$
27: **for** $i = 0$ **to** 12 **do**
28: $\quad c_i \leftarrow u_i \bmod p$
29: **end for**
30: **return** $c = \sum_{i=0}^{12} c_i\omega^i$

3.3 Frobenius map and inversion operation

Table 1. Notation of arithmetic operations in $\mathbb{F}_{p^{13}}$

Notation	Definition
m	Multiplication in $\mathbb{F}_p$
s	Squaring in $\mathbb{F}_p$
a	Addition in $\mathbb{F}_p$
i	Inversion in $\mathbb{F}_p$
r	Modular reduction in $\mathbb{F}_p$
m_u	Multiplication in $\mathbb{F}_p$ without reduction
s_u	Squaring in $\mathbb{F}_p$ without reduction
$\tilde{m}$	Multiplication in $\mathbb{F}_{p^{13}}$
$\tilde{s}$	Squaring in $\mathbb{F}_{p^{13}}$
$\tilde{a}$	Addition in $\mathbb{F}_{p^{13}}$
$\tilde{i}$	Inversion in $\mathbb{F}_{p^{13}}$
$\tilde{i}_c$	Inversion in $\mathbb{G}_{\phi_{13}}$
$\tilde{r}$	Modular reduction in $\mathbb{F}_{p^{13}}$
$\tilde{m}_u$	Multiplication in $\mathbb{F}_{p^{13}}$ without reduction
$\tilde{s}_u$	Squaring in $\mathbb{F}_{p^{13}}$ without reduction
f	Frobenius in $\mathbb{F}_{p^{13}}$
e	Exponentiation by x in $\mathbb{F}_{p^{13}}$
$+$	addition without carry checks
$-$	subtraction without carry checks
$\times$	multiplication without modular reduction
$\oplus$	addition with modular reduction or carry checks
$\ominus$	subtraction with modular reduction or carry checks
$\otimes$	multiplication modular reduction

Let $a = \sum_{i=0}^{12} a_i \omega^i \in \mathbb{F}_{p^{13}}$, where each $a_i \in \mathbb{F}_p$. Since $p \equiv 1 \bmod 13$, it follows that

$$a^{p^j} = (\sum_{i=0}^{12} a_i \omega^i)^{p^j} = \sum_{i=0}^{12} (a_i \cdot \xi^{i \cdot \frac{p^j-1}{13}}) \omega^i.$$

The values of $\xi^{i\frac{p^j-1}{13}}$ can be precomputed for each $i \in \{1, 2, ...12\}$. Thus, one p^i- Frobenius operation in $\mathbb{F}_{p^{13}}$ costs $12m$. Generally, the inversion of $a \in \mathbb{F}_{p^{13}}$ can be computed by using the following formula:

$$a^{-1} = Norm_{\mathbb{F}_{p^{13}}/\mathbb{F}_p}(a)^{-1} \cdot \prod_{\mathrm{i}=1}^{12} a^i, \tag{2}$$

where $Norm_{\mathbb{F}_{p^{13}}/\mathbb{F}_p}(a) = \prod_{i=0}^{12} a^i \in \mathbb{F}_p$. A direct computation of an inversion in $\mathbb{F}_{p^{13}}$ by using Equation (2) requires $12\tilde{m} + 13m + i + 12f$. Now we give another efficient method to compute it. Define $b = \prod_{i=1}^{12} a^i$. Then we find that

$$b = \left((a^p \cdot a^{p^2} \cdot a^{p^3})^{p^3+1}\right)^{p^6+1}. \tag{3}$$

Since $a \cdot b \in \mathbb{F}_p$, it follows that

$$a \times b = a_0 \times b_0 + \xi \cdot (a_1 \times b_{12} + a_2 \times b_{11} + \cdots + a_{12} \times b_1).$$

Hence,

$$a^{-1} = \frac{b}{\left(a_0 \times b_0 + \xi \cdot (a_1 \times b_{12} + \cdots + a_{12} \times b_1)\right) \bmod p}.$$

The total cost is reduced to $4\tilde{m} + 13m + 13m_u + i + 5f + 26a + 1r$. The cyclotomic subgroup $\mathbb{G}_{\Phi_{13}}$ is a subgroup of $\mathbb{F}_{p^{13}}$ such that

$$\mathbb{G}_{\Phi_{13}} = \{a \in \mathbb{F}_{p^{13}} | a^{\Phi_{13}(p)} = 1\}.$$

Compared with the inversion in $\mathbb{F}_{p^{13}}$, an inversion in $\mathbb{G}_{\Phi_{13}}$ is more efficient as $a^{-1} = \prod_{i=1}^{12} a^i$, which requires $4\tilde{m}$ and $5f$ as calculated in (3).

4 Modified Miller double-and-add

In the standard implementation of Miller's algorithm, the Miller loop consists of two parts: doubling (DBL) step and addition (ADD) step . In this section, we give a fast formula by combining one doubling and addition steps into a single step. We name the stage as doubling-addition (DBLADD) step. Generally, the rational function $f_{2m+1,Q}$ can be obtained from $f_{m,Q}$ as follows

$$f_{2m+1,Q} = f_{m,Q}^2 \frac{\ell_{[m]Q,[m]Q} \cdot \ell_{[2m]Q,Q}}{\nu_{[2m]Q} \cdot \nu_{[2m+1]Q}}. \tag{4}$$

When the embedding degree is odd prime, the trick of denominator elimination is not applicable any more. Hence, one doubling-addition step requires four line evaluations in (4). Define the modified Miller function $g_{m,Q}$ to be a normalized rational function such that

$$div(g_{m,Q}) = m(Q) + ([-m]Q) - (m+1)(\mathcal{O}). \tag{5}$$

It follows that

$$g_{1,Q} = x - x_Q, \tag{6}$$

$$g_{2m,Q} = g_{m,Q}^2 \cdot \frac{\nu_{[2m]Q}}{\ell_{[-m]Q,[-m]Q}}, \tag{7}$$

$$g_{m+1,Q} = g_{m,Q} \cdot \frac{\ell_{[m]Q,Q}}{\nu_{[m]Q}}. \tag{8}$$

We also observe that

$$div(g_{u,Q}) - div(f_{u,Q}) = div(\nu_{uQ}).$$

Thus $g_{u,Q} = f_{u,Q} \cdot \nu_{uQ}$ up to $\mathbb{F}_p$. This gives us another way to compute $f_{u,Q}$. In particular, we first compute $g_{u,Q}$ starting from $g_{1,Q}$ by using the iteration relations given in (7)-(8). Then, we recover $f_{u,Q}$ from $g_{u,Q}$ by $f_{uQ} = g_{u,Q}/\nu_{uQ}$. In order to obtain $g_{2m+1,Q}$ from $g_{m,Q}$, it requires to find a rational function with divisor $div(g_{2m+1,Q}) - div(g^2_{m,Q})$. From (5), we immediately have

$$\begin{aligned} &div(g_{2m+1,Q}) - div(g^2_{m,Q}) \\ &=(Q) + ([-2m-1]Q) - 2([-m]Q) \\ &=div(\ell_{[2m]Q,Q}) - div(\ell_{[-m]Q,[-m]Q}) \end{aligned}$$

Hence,

$$g_{2m+1,Q} = g^2_{m,Q} \frac{\ell_{[2m]Q,Q}}{\ell_{[-m]Q,[-m]Q}}. \tag{9}$$

Compared with the formula in (4), it is intuitively plausible that the new formula in (9) would be more efficient, since only two line evaluations are required. In the following, we apply the optimization to the curve arithmetic in Jacobian and homogeneous coordinates. For any integers i and j, we denote by $N_{i,Q}$, $N\ell_{[i]Q,[j]Q}$ and $N\nu_{[i]Q}$ the numerators of $g_{i,Q}(P)$, $\ell_{[i]Q,[j]Q}(P)$ and $\nu_{[i]Q}(P)$, and by $D_{i,Q}$ the denominator of $g_{i,Q}(P)$, respectively.

4.1 Jacobian coordinates

For any point $R \in E$, let x_R and y_R be the x- and y- coordinates of R in affine coordinates. Fast algorithm of elliptic curve scalar multiplication is crucial for pairing computation. Hence, the point T in the subgroup $\mathbb{G}_2$ is generally stored in projective coordinates to avoid the field inversion. For this curve shape, Jacobian coordinates offer the most efficient group operations [30]. In particular, let $T = (X_T, Y_T, Z_T)$ be in Jacobian coordinates. Using the formulas proposed in [31], the point $2T = (X_{2T}, Y_{2T}, Z_{2T})$ is given by :

$$\begin{aligned} X_{2T} &= \frac{9}{4}X_T^4 - 2X_T \cdot Y_T^2, \\ Y_{2T} &= \frac{3}{2}X_T^2 \cdot (X_T \cdot Y_T^2 - X_{2T}) - Y_T^4, \\ Z_{2T} &= Y_T \cdot Z_T. \end{aligned}$$

To compute the above, we use the following sequence of operations:

$$t_0 = X_T^2, t_1 = t_0/2, t_0 = t_0 + t_1, t_1 = t_0^2, t_2 = Y_T^2, t_3 = X_T \cdot t_2,$$

$$X_{2T} = t_1 - 2t_3, t_1 = t_3 - X_{2T}, u_0 = t_0 \times t_1, u_1 = t_2 \times t_2,$$
$$Y_{2T} = (u_0 - u_1) \bmod p, Z_{2T} = Y_T \cdot Z_T,$$

which requires $2\tilde{m} + \tilde{m}_u + 3\tilde{s} + \tilde{s}_u + \tilde{r} + 7\tilde{a}$. It should be noted that the cost of division by two is equivalent to that of addition, and the computation of $u_0 - u_1$ requires $2a$. If $T \neq Q$, then the point $T + Q = (X_{T+Q}, Y_{T+Q}, Z_{T+Q})$ in Jacobian coordinates is given by

$$\begin{aligned}
\alpha_{T+Q} &= y_Q \cdot Z_T^3 - Y_T, \\
\beta_{T+Q} &= x_Q \cdot Z_T^2 - X_T, \\
X_{T+Q} &= \alpha_{T+Q}^2 - 2X_T \cdot \beta_{T+Q}^2 - \beta_{T+Q}^3, \\
Y_{T+Q} &= \alpha_{T+Q} \cdot (X_T \cdot \beta_{T+Q}^2 - X_{T+Q}) - Y_T \cdot \beta_{T+Q}^3, \\
Z_{T+Q} &= Z_T \cdot \beta_{T+Q}.
\end{aligned}$$

We use the following sequence of operations to compute the point addition in $6\tilde{m} + 2\tilde{m}_u + 3\tilde{s} + \tilde{r} + 8\tilde{a}$ as

$$\begin{aligned}
&t_0 = Z_T^2, t_1 = t_0 \cdot Z_T, t_1 = t_1 \cdot y_Q, t_1 = t_1 - Y_T, \\
&t_2 = t_1^2, t_3 = t_0 \cdot x_Q - X_T, t_4 = t_3^2, t_5 = t_3 \cdot t_4, \\
&t_2 = t_2 - t_5, t_4 = t_4 \cdot X_T, X_{T+Q} = t_2 - 2t_4, \\
&t_4 = t_4 - X_{T+Q}, u_0 = t_1 \times t_4, u_1 = Y_T \times t_5, \\
&Y_{T+Q} = (u_0 - u_1) \bmod p, Z_{T+Q} = Z_T \cdot t_3.
\end{aligned}$$

doubling step According to the new iteration formula in (7), we require to compute $\ell_{-T,-T}(P)$ and $\nu_{2T}(P)$ at doubling step. Logically, we first compute $2T$ using one point doubling as proposed above. Note that $\ell_{-T,-T}$ can be written as two different formulas, that is,

$$\begin{aligned}
y &= -\frac{3x_T^2}{2y_T}(x - x_T) - y_T, \\
y &= -\frac{3x_T^2}{2y_T}(x_P - x_{2T}) + y_{2T}.
\end{aligned}$$

In our experience, it is more efficient to select the latter one at the step, since it shares more common intermediate values with $\nu_{2T}(P)$. Specifically, the line functions $\ell_{-T,-T}$ and ν_{2T} evaluated at $P = (x_P, y_P)$ in Jacobian coordinates can be expressed as

$$\begin{aligned}
\ell_{-T,-T}(P) &= \frac{y_P \cdot Z_{2T}^3 + \frac{3}{2}X_T^2 \cdot (x_P \cdot Z_{2T}^2 - X_{2T}) - Y_{2T}}{Z_{2T}^3}, \\
\nu_{2T}(P) &= \frac{x_P \cdot Z_{2T}^2 - X_{2T}}{Z_{2T}^2}.
\end{aligned}$$

Then, the values of $N_{2m,Q}$ and $D_{2m,Q}$ are given by

$$N_{2m,Q} = N_{m,Q}^2 \cdot Z_{2T} \cdot N\nu_{2T},$$

$$D_{2m,Q} = D_{m,Q}^2 \cdot N\ell_{-T,-T}.$$

Note that $\frac{3}{2}X_T^2$ has been given in the point doubling step. Hence, the values of $N_{2m,Q}$ and $D_{2m,Q}$ can be computed by using the following sequence of operations:

$$\begin{aligned}
&t_0 = Z_{2T}^2, t_1 = t_0 \cdot Z_{2T}, t_0 = x_P \cdot t_0, t_0 = t_0 - X_{2T},\\
&u_0 = y_P \times t_1, t_2 = Z_{2T} \cdot t_0, t_3 = N_{m,Q}^2, N_{2m,Q} = t_2 \cdot t_3,\\
&u_1 = \frac{3}{2}X_T^2 \times t_0,\ t_0 = (u_0 + u_1) \bmod p,\ t_0 = t_0 - Y_{2T},\\
&t_1 = D_{m,Q}^2, D_{2m,Q} = t_0 \cdot t_1,
\end{aligned}$$

which comes at a cost of $4\tilde{m} + \tilde{m}_u + 3\tilde{s} + 13m + 13m_u + \tilde{r} + 4\tilde{a}$. In total, the number of operations required at the doubling step is $6\tilde{m} + 2\tilde{m}_u + 6\tilde{s} + \tilde{s}_u + 13m + 13m_u + 2\tilde{r} + 11\tilde{a}$.

addition step We first compute $T + Q$ by using one point addition. Then the two line functions $\ell_{T,Q}$ and ν_T evaluated at $P = (x_P, y_P)$ in Jacobian coordinates are given by:

$$\begin{aligned}
\ell_{T,Q}(P) &= \frac{\beta_{T+Q} \cdot (y_P \cdot Z_T^3 - Y_T) - \alpha_{T+Q}(x_P \cdot Z_T^2 - X_T)}{Z_{T+Q} \cdot Z_T^2},\\
\nu_T(P) &= \frac{x_P \cdot Z_T^2 - X_T}{Z_T^2},
\end{aligned}$$

where $\alpha_{T+Q}, \beta_{T+Q}, Z_T^2$ and Z_T^3 have been given in the point addition step. From (8), the values of $N_{m+1,Q}$ and $D_{m+1,Q}$ are given by

$$\begin{aligned}
N_{m+1,Q} &= N_{m,Q} \cdot N\ell_{T,Q},\\
D_{m+1,Q} &= D_{m,Q} \cdot Z_{T+Q} \cdot N\nu_T.
\end{aligned}$$

To compute $N_{m+1,Q}$ and $D_{m+1,Q}$, we use the following sequence of operations:

$$\begin{aligned}
&t_0 = y_P \cdot Z_T^3 - Y_T, t_1 = x_P \cdot Z_T^2 - X_T, u_0 = \beta_{T+Q} \times t_0,\\
&u_1 = \alpha_{T+Q} \times t_1, t_0 = (u_0 - u_1) \bmod p, t_1 = t_1 \cdot Z_{T+Q},\\
&N_{m+1,Q} = N_{m,Q} \cdot t_0, D_{m+1,Q} = D_{m,Q} \cdot t_1,
\end{aligned}$$

which comes at a cost of $3\tilde{m} + 2\tilde{m}_u + 26m + \tilde{r} + 4\tilde{a}$. Hence, the number of operations required at the addition step is $9\tilde{m} + 4\tilde{m}_u + 3\tilde{s} + 26m + 2\tilde{r} + 12\tilde{a}$.

doubling-addition step The point $2T + Q$ in Jacobian coordinates can be obtained by performing one point doubling and one point addition. Consequently, the line functions $\ell_{2T,Q}$ and $\ell_{-T,-T}$ evaluated at $P = (x_P, y_P)$ are given by

$$\begin{aligned}
\delta_1 &= \beta_{2T+Q} \cdot (y_P \cdot Z_{2T}^3 - Y_{2T}),\\
\delta_2 &= x_P \cdot Z_{2T}^2 - X_{2T},
\end{aligned}$$

$$\ell_{2T,Q}(P) = \frac{\delta_1 - \alpha_{2T+Q} \cdot \delta_2}{\beta_{2T+Q} \cdot Z_{2T}^3},$$
$$\ell_{-T,-T}(P) = \frac{\delta_1 + \frac{3}{2}X_T^2 \cdot \beta_{2T+Q} \cdot \delta_2}{\beta_{2T+Q} \cdot Z_{2T}^3},$$

where $\frac{3}{2}X_T^2$, α_{2T+Q}, β_{2T+Q}, Z_{2T}^2 and Z_{2T}^3 have been given in point doubling or addition steps. The numerator and denominator of $\ell_{-T,-T}(P)$ are multiplied by β_{2T+Q} simultaneously. By this way, one modular reduction in $\mathbb{F}_{p^{13}}$ can be saved as lazy reduction can be used when computing the numerator of $\ell_{-T,-T}(P)$. From (9), the values of $N_{2m+1,Q}$ and $D_{2m+1,Q}$ are given by

$$N_{2m+1,Q} = N_{m,Q}^2 \cdot N\ell_{2T,Q},$$
$$D_{2m+1,Q} = D_{m,Q}^2 \cdot N\ell_{-T,-T}.$$

We use the following sequence of operations to compute $N_{2m+1,Q}$ and $D_{2m+1,Q}$:

$$t_0 = y_P \cdot Z_{2T}^3 - Y_{2T}, u_0 = t_0 \times \beta_{2T+Q}, t_0 = x_P \cdot Z_{2T}^2,$$
$$t_0 = t_0 - X_{2T}, u_1 = t_0 \times \alpha_{2T+Q}, t_1 = (u_0 - u_1) \bmod p,$$
$$t_2 = N_{m,Q}^2, N_{2m+1,Q} = t_1 \cdot t_2, t_1 = \frac{3}{2}X_T^2 \cdot \beta_{2T+Q}, u_1 = t_0 \times t_2,$$
$$t_0 = (u_0 + u_1) \bmod p, t_1 = D_{m,Q}^2, D_{2m+1,Q} = t_0 \cdot t_1,$$

which comes at a cost of $3\tilde{m} + 3\tilde{m}_u + 2\tilde{s} + 26m + 2\tilde{r} + 6\tilde{a}$. Thus, the total number of operations required at the doubling-addition step is $11\tilde{m} + 6\tilde{m}_u + 8\tilde{s} + \tilde{s}_u + 26m + 4\tilde{r} + 21\tilde{a}$.

4.2 Homogeneous coordinates

Homogeneous coordinates offer another efficient way to implement elliptic curve scalar multiplication. Compared with Jacobian coordinates, Azarderakhsh *et al*. [31] examine that homogeneous coordinates are the preferred choice for the optimal ate pairing implementation over BN curves. In particular, let $T = (X_T, Y_T, Z_T) \in E(F_{p^{13}})$ in homogeneous coordinates. Then the point $2T = (X_{2T}, Y_{2T}, Z_{2T})$ in homogeneous coordinates is given as follows:

$$X_{2T} = \frac{X_T \cdot Y_T}{2} \cdot (Y_T^2 + 9bZ_T^2),$$
$$Y_{2T} = \left(\frac{Y_T^2 - 9bZ_T^2}{2}\right)^2 - 27b^2Z_T^4,$$
$$Z_{2T} = 2Y_T^3 \cdot Z_T.$$

We use the following sequence of operations to compute the point doubling in $3\tilde{m} + 3\tilde{s} + 2\tilde{s}_u + \tilde{r} + 22\tilde{a}$ as

$$t_0 = \frac{X_T \cdot Y_T}{2},\ t_1 = Y_T^2,\ t_2 = Z_T^2,\ t_3 = 3bt_2,\ t_4 = 3t_3,$$

$$t_5 = t_1 + t_4, X_{2T} = t_0 \cdot t_5, t_5 = (Y_T + Z_T)^2 - t_1 - t_2,$$
$$Z_{2T} = t_1 \cdot t_5, t_0 = \frac{t_1 - t_4}{2}, u_0 = t_0 \times t_0, u_1 = t_3 \times t_3,$$
$$Y_{2T} = (u_0 - 3u_1) \bmod p,$$

where b is the curve parameter. If $T \neq Q$, the point $T + Q = (X_{T+Q}, Y_{T+Q}, Z_{T+Q})$ in homogeneous coordinates is given by

$$\begin{aligned}
\alpha_{T+Q} &= Y_T - y_Q \cdot Z_T, \\
\beta_{T+Q} &= X_T - x_Q \cdot Z_T, \\
X_{T+Q} &= \beta_{T+Q}(\beta_{T+Q}^3 + Z_T \cdot \alpha_{T+Q}^2 - 2X_T \cdot \beta_{T+Q}^2), \\
Y_{T+Q} &= \alpha_{T+Q}(3X_T \cdot \beta_{T+Q}^2 - \beta_{T+Q}^3 - Z_T \cdot \alpha_{T+Q}^2) - Y_T \cdot \beta_{T+Q}^3, \\
Z_{T+Q} &= Z_T \cdot \beta_{T+Q}^3.
\end{aligned}$$

We compute the point $T + Q$ using the following sequence of operations:

$$\begin{aligned}
&t_0 = Y_T - y_Q \cdot Z_T,\ t_1 = t_0^2,\ t_2 = X_T - x_Q \cdot Z_T,\ t_3 = t_2^2, \\
&t_4 = t_2 \cdot t_3, \quad t_5 = t_1 \cdot Z_T + t_4, \quad t_6 = X_T \cdot t_3, \quad t_7 = 2t_6, \\
&t_5 = t_5 - t_7, \quad X_{T+Q} = t_2 \cdot t_5, \quad t_7 = t_6 - t_5, \quad u_0 = t_0 \times t_7, \\
&u_1 = Y_T \times t_4, Y_{T+Q} = (u_0 - u_1) \bmod p, Z_{T+Q} = Z_T \cdot t_4.
\end{aligned}$$

The total cost of point addition in homogeneous coordinates is $7\tilde{m} + 2\tilde{m}_u + 2\tilde{s} + \tilde{r} + 8\tilde{a}$.

doubling step Unlike the case of the doubling step in Jacobian coordinates, it is appropriate to select the formula of $\ell_{-T,-T}$ as

$$y = -\frac{3x_T^2}{2y_T}(x - x_T) - y_T.$$

This is mainly due to the curve equation (1) can be used to speed up line evaluation in this situation. Indeed, the value of X_T^3/Z_T can be replaced by $Y_T^2 + bZ_T^2$ in homogeneous coordinates. As a result, the tangent line $\ell_{-T,-T}$ and vertical line ν_{2T} evaluated at $P = (x_P, y_P)$ are given by

$$\begin{aligned}
\ell_{-T,-T}(P) &= \frac{2Y_T Z_T \cdot y_P + 3X_T^2 \cdot x_P - Y_T^2 - 3bZ_T^2}{2Y_T Z_T}, \\
\nu_{2T}(P) &= \frac{x_P \cdot Z_{2T} - X_{2T}}{Z_{2T}},
\end{aligned}$$

where $2Y_T Z_T, Y_T^2$ and $3bZ_T^2$ have been given in point doubling step. Therefore, the values of $N_{2m,Q}$ and $D_{2m,Q}$ are given by

$$N_{2m,Q} = N_{m,Q}^2 \cdot N\nu_{2T},$$

$$D_{2m,Q} = D_{m,Q}^2 \cdot Y_T^2 \cdot N\ell_{-T,-T}.$$

We compute $N_{2m,Q}$ and $D_{2m,Q}$ by using the following sequence of operations in $3\tilde{m} + 3\tilde{s} + 13m + 26m_u + \tilde{r} + 7\tilde{a}$ as

$$\begin{aligned}
&t_0 = x_P \cdot Z_{2T} - X_{2T}, t_1 = N_{m,Q}^2, N_{2m,Q} = t_0 \cdot t_1,\\
&u_0 = y_P \times 2Y_T Z_T, t_0 = 3X_T^2, u_1 = x_P \times t_0,\\
&t_0 = (u_0 + u_1) \bmod p, t_0 = t_0 - Y_T^2 - 3bZ_T^2,\\
&t_0 = t_0 \cdot Y_T^2, t_1 = D_{m,Q}^2, D_{2m,Q} = t_0 \cdot t_1.
\end{aligned}$$

Hence, the number of operations required in the doubling step is $6\tilde{m} + 6\tilde{s} + 2\tilde{s}_u + 13m + 26m_u + 2\tilde{r} + 29\tilde{a}$.

addition step The point $T + Q$ can be obtained by using one point addition step. The two line functions $\ell_{T,Q}$ and ν_T evaluated at $P = (x_P, y_P)$ are given by:

$$\begin{aligned}
\ell_{T,Q}(P) &= \frac{\beta_{T+Q} \cdot (y_p \cdot Z_T - Y_T) - \alpha_{T+Q} \cdot (x_P \cdot Z_T - X_T)}{\beta_{T+Q} \cdot Z_T},\\
\nu_T(P) &= \frac{x_P \cdot Z_T - X_T}{Z_T},
\end{aligned}$$

where α_{T+Q} and β_{T+Q} have been given in the point addition step. As a result, the values of $N_{m+1,Q}$ and $D_{m+1,Q}$ are given by

$$\begin{aligned}
N_{m+1,Q} &= N_{m,Q} \cdot N\ell_{T,Q},\\
D_{m+1,Q} &= D_{m,Q} \cdot \beta_{T+Q} \cdot N\nu_T.
\end{aligned}$$

The following sequence of operations can be used to compute the values $N_{m+1,Q}$ and $D_{m+1,Q}$:

$$\begin{aligned}
&t_0 = y_p \cdot Z_T - Y_T,\ t_1 = x_P \cdot Z_T - X_T,\ u_0 = t_0 \times \beta_{T+Q},\\
&u_1 = t_1 \times \alpha_{T+Q}, t_0 = (u_0 - u_1) \bmod p, N_{m+1,Q} = t_0 \cdot N_{m,Q},\\
&t_1 = t_1 \cdot D_{m,Q}, D_{m+1,Q} = \beta_{T+Q} \cdot t_1,
\end{aligned}$$

which comes at a cost of $3\tilde{m} + 2\tilde{m}_u + 26m + \tilde{r} + 4\tilde{a}$. Hence, the total number of operations required at the addition step is $10\tilde{m} + 4\tilde{m}_u + 2\tilde{s} + 26m + 2\tilde{r} + 12\tilde{a}$.

doubling-addition step We obtain the point $2T + Q$ by using one point doubling and one point addition. The line functions $\ell_{-T,-T}$ and $\ell_{2T,Q}$ evaluated at $P = (x_P, y_P)$ are given by

$$\ell_{2T,Q}(P) = \frac{\beta_{2T+Q} \cdot (y_P - y_Q) - \alpha_{2T+Q} \cdot (x_P - x_Q)}{\beta_{2T+Q}},$$

$$\ell_{-T,-T}(P) = \frac{2Y_T Z_T \cdot y_P + 3X_T^2 \cdot x_P - Y_T^2 - 3bZ_T^2}{2Y_T Z_T},$$

where $\alpha_{2T+Q}, \beta_{2T+Q}, 2Y_T Z_T, Y_T^2$ and $3bZ_T^2$ have been given in the point doubling or addition steps. Thus, the values of $N_{2m+1,Q}$ and $D_{2m+1,Q}$ are given by

$$N_{2m+1,Q} = N_{m,Q}^2 \cdot 2Y_T Z_T \cdot N\ell_{2T,Q},$$
$$D_{2m+1,Q} = D_{m,Q}^2 \cdot \beta_{2T+Q} \cdot N\ell_{-T,-T}.$$

To compute $N_{2m+1,Q}$ and $D_{2m+1,Q}$, we use the following sequence of operations:

$$u_0 = \beta_{2T+Q} \times (y_P - y_Q), \quad u_1 = \alpha_{2T+Q} \times (x_P - x_Q),$$
$$t_0 = (u_0 - u_1) \bmod p, t_0 = t_0 \cdot 2Y_T Z_T, N_{2m+1,Q} = t_0 \cdot N_{m,Q}^2,$$
$$t_0 = 3X_T^2, u_0 = 2Y_T Z_T \times y_P, u_1 = t_0 \times x_P, t_0 = (u_0 + u_1) \bmod p,$$
$$t_0 = t_0 - Y_T^2 - 3bZ_T^2, t_0 = t_0 \cdot \beta_{2T+Q}, D_{2m+1,Q} = t_0 \cdot D_{m,Q}^2.$$

Hence, it costs $4\tilde{m} + 2\tilde{m}_u + 3\tilde{s} + 26m_u + 2\tilde{r} + 10\tilde{a}$. In total, the number of operations required at the doubling-addition step is $14\tilde{m} + 4\tilde{m}_u + 8\tilde{s} + 2\tilde{s}_u + 26m_u + 4\tilde{r} + 40\tilde{a}$.

4.3 Comparison of operation counts

In Table 2, we draw a comparison of the operation counts of every stage in the modified Miller double-and-add between Jacobian and homogeneous coordinates. The operation counts using the method proposed in [19] is also listed. We ignore addition and the optimization by the technique of lazy reduction. The results show that homogeneous coordinates are the best choice, since it offers the fastest formula at doubling step. Moreover, one doubling-addition step requires less computation than the sum of one doubling and one addition steps in both homogeneous and Jacobian coordinates. In particular, if we assume that $\tilde{m} \approx 0.8\tilde{s}$ and $\tilde{m} \approx 66m$ (see Table 4), one doubling-addition step is about 8.9% and 17.4% faster than the sum of one doubling and one addition steps in homogeneous and Jacobian coordinates, respectively.

Table 2. Operation counts for the iteration of Miller double-and-add in different way(ignoring addition and the optimization of lazy reduction)

Coordinates	DBL	ADD	DBLADD
Jacobian [19]	$9\tilde{m} + 7\tilde{s} + 39m$	$14\tilde{m} + 3\tilde{s} + 13m$	$23\tilde{m} + 10\tilde{s} + 52m$
Jacobian (this work)	$8\tilde{m} + 7\tilde{s} + 26m$	$13\tilde{m} + 3\tilde{s} + 26m$	$17\tilde{m} + 9\tilde{s} + 26m$
Homogeneous (this work)	$6\tilde{m} + 8\tilde{s} + 39m$	$14\tilde{m} + 2\tilde{s} + 26m$	$18\tilde{m} + 10\tilde{s} + 26m$

5 Modified Miller quadruple-and-add

Inspired by the idea proposed in [32], we combine two modified doubling steps into one modified quadrupling (QPL) step. In the classic Miller's algorithm, the modified Miller function $g_{4m,Q}$ can be obtained from $g_{m,Q}$ by using two consecutive doubling steps, that is,

$$g_{4m,Q} = g_{m,Q}^4 \cdot \frac{\nu_{[2m]Q}^2 \cdot \nu_{[4m]Q}}{\ell_{[-m]Q,[-m]Q}^2 \cdot \ell_{[-2m]Q,[-2m]Q}} \tag{10}$$

The difference between the $div(g_{4m,Q})$ and $div(g_{m,Q}^4)$ are given by

$$\begin{aligned} &div(g_{4m,Q}) - div(g_{m,Q}^4) \\ =&([-4m]Q) - 4([-m]Q) + 3(\mathcal{O}) \\ =&div(\ell_{[2m]Q,[2m]Q}) - div(\ell_{[-m]Q,[-m]Q}^2). \end{aligned}$$

Hence,

$$g_{4m,Q} = g_{m,Q}^4 \frac{\ell_{[2m]Q,[2m]Q}}{\ell_{[-m]Q,[-m]Q}^2}. \tag{11}$$

As compared to (10), the number of line functions required in Miller function update is reduced to two in (11). The above optimization is analogous to the trick of denominator elimination for pairing computation with even embedding degree, as both of them eliminate vertical line valuation.

5.1 Jacobian coordinates

quadrupling step The point $4T = (X_{4T}, Y_{4T}, Z_{4T})$ in Jacobian coordinates can be obtained by computing two consecutive point doublings. Then, the tangent lines $\ell_{2T,2T}$ and $\ell_{-T,-T}$ evaluated at P are given by

$$\begin{aligned} \delta_1 &= y_P \cdot Z_{4T} \cdot Z_{2T}^2, \\ \delta_2 &= x_P \cdot Z_{2T}^2 - X_{2T}, \\ \ell_{2T,2T}(P) &= \frac{\delta_1 - \frac{3}{2}X_{2T}^2 \cdot \delta_2 - Y_{2T}^2}{Z_{4T} \cdot Z_{2T}^2}, \\ \ell_{-T,-T}(P) &= \frac{\delta_1 + \frac{3}{2}X_T^2 \cdot Y_{2T} \cdot \delta_2 - Y_{2T}^2}{Z_{4T} \cdot Z_{2T}^2}, \end{aligned}$$

where $\frac{3}{2}X_T^2$, $\frac{3}{2}X_{2T}^2$ and Y_{2T}^2 have been given in point doubling steps. Analogous to the doubling-addition step, the numerator and denominator of $\ell_{-T,-T}(P)$ are multiplied by Y_{2T} simultaneously. According to (11), the values of $N_{4m,Q}$ and $D_{4m,Q}$ are given by

$$N_{4m,Q} = N_{m,Q}^4 \cdot Z_{4T} \cdot Z_{2T}^2 \cdot N\ell_{2T,2T},$$

$$D_{4m,Q} = D_{m,Q}^4 \cdot (N\ell_{-T,-T})^2.$$

We use the following sequence of operations to compute $N_{4m,Q}$ and $D_{4m,Q}$:

$$\begin{aligned}
&t_0 = Z_{2T}^2, t_1 = x_P \cdot t_0 - X_{2T}, t_0 = t_0 \cdot Z_{4T}, u_0 = y_P \times t_0,\\
&u_1 = \frac{3}{2} X_{2T}^2 \times t_1, t_2 = (u_0 - u_1) \bmod p, t_2 = t_2 - Y_{2T}^2,\\
&t_3 = t_0 \cdot t_2, t_0 = N_{m,Q}^2, N_{4m,Q} = t_0^2 \cdot t_3, t_0 = \frac{3}{2} X_T^2 \cdot Y_{2T},\\
&u_2 = t_0 \times t_1, t_0 = (u_0 + u_2) \bmod p, t_0 = t_0 - Y_{2T}^2, t_1 = D_{m,Q}^2,\\
&t_1 = t_0 \cdot t_1, D_{4m,Q} = t_1^2,
\end{aligned}$$

which comes at a cost of $5\tilde{m} + 2\tilde{m}_u + 5\tilde{s} + 13m + 13m_u + 2\tilde{r} + 7\tilde{a}$. As a consequence, the total number of operations required at the quadrupling step is $9\tilde{m} + 4\tilde{m}_u + 11\tilde{s} + 2\tilde{s}_u + 13m + 13m_u + 4\tilde{r} + 21\tilde{a}$.

quadrupling-addition step Define quadrupling-addition (QPLADD) to be the step that obtaining $g_{4m+1,Q}$ from $g_{m,Q}$. There are two feasible schemes that need to be considered for performing the step. At the first scheme, this step can be decomposed as one quadrupling and one addition steps. According to the above analysis, the total number of operation counts required in this scheme is $18\tilde{m} + 8\tilde{m}_u + 14\tilde{s} + 2\tilde{s}_u + 39m + 13\tilde{m}_u + 6\tilde{r} + 33\tilde{a}$. At the second scheme, this step can be decomposed as one doubling and one doubling-addition steps. The total number of operation counts required in this scheme is $17\tilde{m} + 8\tilde{m}_u + 14\tilde{s} + 2\tilde{s}_u + 39m + 13m_u + 6\tilde{r} + 32\tilde{a}$. We select the second scheme to execute quadrupling-addition step since it saves $\tilde{m} + \tilde{a}$ compared to the first one.

quadrupling-doubling step Define quadrupling-doubling (QPLDBL) to be the step that obtaining $g_{4m+2,Q}$ from $g_{m,Q}$. At the step, the only scheme is to perform one doubling-addition step and one doubling step. Hence, the total number required in this step is equal to that in quadrupling-addition step.

5.2 Homogeneous coordinates

quadrupling step In homogeneous coordinates, the two tangent lines $\ell_{2T,2T}$ and $\ell_{-T,-T}$ evaluated at P are given by

$$\begin{aligned}
\ell_{2T,2T}(P) &= \frac{2Y_{2T}Z_{2T} \cdot y_P - 3X_{2T}^2 \cdot x_P + Y_{2T}^2 + 3bZ_{2T}^2}{2Y_{2T}Z_{2T}},\\
\ell_{-T,-T}(P) &= \frac{2Y_T Z_T \cdot y_P + 3X_T^2 \cdot x_P - Y_T^2 - 3bZ_T^2}{2Y_T Z_T},
\end{aligned}$$

where $2Y_T Z_T, 2Y_{2T}Z_{2T}, Y_T^2, Y_{2T}^2, 3bZ_T^2$ and $3bZ_{2T}^2$ have been given in the point doubling steps. According to Formula (11), we have

$$N_{4m,Q} = N_{m,Q}^4 \cdot (2Y_T Z_T)^2 \cdot N\ell_{2T,2T},$$

$$D_{4m,Q} = D_{m,Q}^4 \cdot (N\ell_{-T,-T})^2 \cdot 2Y_{2T}Z_{2T}.$$

We use the following sequence of operations to compute $N_{4m,Q}$ and $D_{4m,Q}$:

$$\begin{aligned}
&t_0 = 3X_{2T}^2, u_0 = 2Y_{2T}Z_{2T} \times y_P, u_1 = t_0 \times x_P,\\
&t_0 = (u_0 - u_1) \bmod p,\ t_0 = t_0 + Y_{2T}^2 + 3bZ_{2T}^2,\\
&t_1 = N_{2m,Q}^2, t_1 = t_1 \cdot 2Y_T Z_T, N_{4m,Q} = t_0 \cdot t_1^2, t_0 = 3X_T^2,\\
&u_0 = 2Y_T Z_T \times y_P, u_1 = t_0 \times x_P, t_0 = (u_0 + u_1) \bmod p,\\
&t_0 = t_0 - Y_T^2 - 3bZ_T^2, t_1 = D_{m,Q}^2, t_1 = t_1 \cdot t_0, t_1 = t_1^2,\\
&D_{4m,Q} = t_1 \cdot 2Y_{2T}Z_{2T},
\end{aligned}$$

which comes at a cost of $4\tilde{m} + 6\tilde{s} + 52m_u + 2\tilde{r} + 12\tilde{a}$. As a consequence, the number of operations required in modified Miller function update at the quadrupling step is $10\tilde{m} + 12\tilde{s} + 4\tilde{s}_u + 52m_u + 4\tilde{r} + 56\tilde{a}$.

quadrupling-addition step At the quadrupling-addition step, the results show the cost of the two proposed schemes are identical, that is, $20\tilde{m}+4\tilde{m}_u+14\tilde{s}+4\tilde{s}_u+52m_u+6\tilde{r}+69\tilde{a}$. So does the quadrupling-doubling step.

5.3 Comparison and optimal strategy

In table 3, we list the operation counts of every stage in the modified Miller quadruple-and-add. The results show that the costs of one quadrupling step are almost identical in the two projective coordinates. However, Jacobian coordinates is preferred for both quadrupling-addition and quadrupling-doubling steps. Hence, we conclude that Jacobian coordinates are the faster choice. In the following, we only implement pairing computation in Jacobian coordinates. Moreover, as compared to the data in Table 2, one quadrupling step is about 15.0% faster than two consecutive doubling steps in Jacobian coordinates. In conclusion, the optimal strategy to implement Miller's algorithm in modified Miller quadruple-and-add is as follows: (1) two consecutive doubling steps are combined into one quadrupling step; (2) one doubling and one addition steps are combined into one doubling-addition step.

Table 3. Operation counts for the iteration of Miller quadruple-and-add in different coordinates systems(ignoring addition and the optimization of lazy reduction)

Coordinates	QPL	QPLADD	QPLDBL
Jacobian	$13\tilde{m} + 13\tilde{s} + 26m$	$25\tilde{m} + 16\tilde{s} + 52m$	$25\tilde{m} + 16\tilde{s} + 52m$
Homogeneous	$10\tilde{m} + 16\tilde{s} + 52m$	$24\tilde{m} + 18\tilde{s} + 52m$	$24\tilde{m} + 18\tilde{s} + 52m$

6 Final Exponentiation

By using the method proposed in Section 2.2, the exponent $d = (p^{13} - 1)/r$ can be equivalently expressed as

$$d = (p-1) \cdot \sum_{i=0}^{11} t_i \cdot p^i,$$

where the coefficients t_i for $i \in \{0, 1, \cdots, 11\}$ are given by

$$\begin{aligned}
t_0 =& -x^{16} - 2x^{15} - 3x^{14} - 2x^{13} - x^{12} + 3x^2 + 3x + 3,\\
t_1 =& -x^{27} - 2x^{26} - 2x^{25} - x^{24} - x^{16} - x^{15} - 2x^{14} +\\
& 2x^{13} + 2x^{12} + 4x^2 + x + 4,\\
t_2 =& x^{25} + x^{24} + x^{23} - x^{16} - x^{15} - x^{14} + 2x^{12} - x^{11} +\\
& x^{10} + 4x^2 + x + 1,\\
t_3 =& x^{25} + x^{24} + x^{23} - x^{16} - x^{15} - x^{14} - 4x^{11} - x^{10} -\\
& x^9 + 4x^2 + x + 1,\\
t_4 =& -x^{24} - 2x^{23} - 2x^{22} - x^{21} - x^{16} - x^{15} - x^{14} - x^{11} +\\
& 2x^{10} + 2x^9 + 4x^2 + x + 1,\\
t_5 =& x^{22} + x^{21} + x^{20} - x^{16} - x^{15} - x^{14} + 2x^9 - x^8 + x^7\\
& + 4x^2 + x + 1,\\
t_6 =& x^{22} + x^{21} + x^{20} - x^{16} - x^{15} - x^{14} - 4x^8 - x^7 - x^6\\
& + 4x^2 + x + 1,\\
t_7 =& -x^{21} - 2x^{20} - 2x^{19} - x^{18} - x^{16} - x^{15} - x^{14} - x^8\\
& + 2x^7 + 2x^6 + 4x^2 + x + 1,\\
t_8 =& x^{19} + x^{18} + x^{17} - x^{16} - x^{15} - x^{14} + 2x^6 - x^5 + x^4\\
& + 4x^2 + x + 1,\\
t_9 =& x^{19} + x^{18} + x^{17} - x^{16} - x^{15} - x^{14} - 4x^5 - x^4 -\\
& x^3 + 4x^2 + x + 1,\\
t_{10} =& -x^{18} - 2x^{17} - 3x^{16} - 2x^{15} - x^{14} - x^5 + 2x^4 +\\
& 2x^3 + 4x^2 + x + 1,\\
t_{11} =& x^4 + 2x^3 + 3x^2 + 2x + 1.
\end{aligned}$$

Denote Nf and Df by the numerator and denominator of $f_{x^2,Q} \cdot f_{x,Q}^p \cdot \ell_{\pi^2(Q),\pi([x]Q)}$, respectively. We first compute $f = (Nf/Df)^{p-1}$, which totally costs $3\tilde{m} + \tilde{i} + 2f$. Then, we raise f to the power $\sum_{i=0}^{11} t_i \cdot p^i$. It seems costly as the relations among t_i are chaotic. By using the following lemma, we fortunately find some relations among t_i.

Lemma 1. *Let k be a prime number, f any of an element in $\mathbb{G}_{\Phi_k}$ and $\frac{\Phi_k(p)}{r} = \sum_{i=0}^{k-2} t_i \cdot p^i$. Then we have*

$$f^{\sum_{i=0}^{k-2} t_i \cdot p^i} = f^{\sum_{i=0}^{k-2} (t_i + c) \cdot p^i + c \cdot p^{k-1}}$$

for any $c \in \mathbb{Z}$.

Proof. Since $f \in \mathbb{G}_{\Phi_k}$, we have $f^{\sum_{i=1}^{k-1} p^i} = 1$. For any $c \in \mathbb{Z}$, it is easy to see $f^{\sum_{i=1}^{k-1} c \cdot p^i} = 1$. It implies that

$$f^{\sum_{i=0}^{k-2} (t_i+c)\cdot p^i + c\cdot p^{k-1}} = f^{\sum_{i=0}^{k-2} t_i \cdot p^i} \cdot f^{\sum_{i=0}^{k-1} c\cdot p^i} = f^{\sum_{i=0}^{k-2} t_i \cdot p^i}.$$

□

Define $c = \lambda_{12} = x^{16} + x^{15} + x^{14} - 4x^2 - x - 1$ and $\lambda_i = t_i + c$ for $i \in \{1, 2, \cdots, 11\}$. Then, it follows that

$$\begin{aligned}
&\lambda_0 = -x^{15} - 2x^{14} - 2x^{13} - x^{12} - x^2 + 2x + 2,\\
&\lambda_{10} = -x^{18} - 2x^{17} - 2x^{16} - x^{15} - x^5 + 2x^4 + 2x^3,\\
&\lambda_{11} = x^{16} + x^{15} + x^{14} + x^4 + 2x^3 - x^2 + x,\\
&\lambda_{12} = c = x^{16} + x^{15} + x^{14} - 4x^2 - x - 1,\\
&\lambda_1 = \lambda_{10} \cdot x^9 + 3, \lambda_2 = \lambda_{11} \cdot x^9, \lambda_3 = \lambda_{12} \cdot x^9,\\
&\lambda_4 = \lambda_{10} \cdot x^6, \quad \lambda_5 = \lambda_{11} \cdot x^6, \quad \lambda_6 = \lambda_{12} \cdot x^6,\\
&\lambda_7 = \lambda_{10} \cdot x^3, \quad \lambda_8 = \lambda_{11} \cdot x^3, \quad \lambda_9 = \lambda_{12} \cdot x^3.
\end{aligned}$$

From Lemma 1, the exponent $\sum_{i=0}^{11} t_i \cdot p^i$ can be replaced by $\sum_{i=0}^{12} \lambda_i \cdot p^i$, which is rewritten as

$$\sum_{i=0}^{12} \lambda_i \cdot p^i = \lambda_0 + 3 \cdot p + \Big(\sum_{i=0}^{2} \lambda_{10+i} \cdot p^i\Big) \cdot \Big(\sum_{i=0}^{3} x^{3i} \cdot p^{10-3i}\Big).$$

In order to raise f to the power of $\sum_{i=0}^{12} \lambda_i \cdot p^i$, we first compute $v_1 = f^{\lambda_0}$ and $v_2 = f^{\lambda_{10}+\lambda_{11}\cdot p+\lambda_{12}\cdot p^2}$ by using the following formula:

$$\begin{aligned}
&g_1 \leftarrow f^x, \ \ g_2 \leftarrow g_1^x, \ \ g_3 \leftarrow g_2^x, \ \ g_4 \leftarrow g_3^x, \ \ g_5 \leftarrow g_4^x,\\
&d_1 \leftarrow f \cdot g_1, d_2 \leftarrow g_1 \cdot g_3, d_3 \leftarrow d_1 \cdot g_2^4, d_4 \leftarrow g_3 \cdot g_4,\\
&h_1 \leftarrow (g_5 \cdot d_4)^{x^9}, \quad h_2 \leftarrow h_1^x, \quad h_3 \leftarrow h_2^x, \quad h_4 \leftarrow h_3^x,\\
&h_5 \leftarrow \ h_4^x, r_1 \leftarrow h_1 \cdot h_2, r_2 \leftarrow h_4 \cdot h_5 \cdot g_5, r_3 \leftarrow h_3 \cdot d_2,\\
&r_3 \leftarrow r_3 \cdot d_4, \ v_1 \leftarrow \frac{d_1^2}{r_1}, \ v_2 \leftarrow d_4^2 \cdot r_3^p, \ v_2 \leftarrow v_2 \cdot h_4^{p^2},\\
&v_3 \leftarrow r_2 \cdot g_2^p, v_3 \leftarrow v_3 \cdot d_3^{p^2}, v_2 \leftarrow \frac{v_2}{v_3}.
\end{aligned}$$

Notice that the above inversion is in the cyclotomic subgroup $\mathbb{G}_{\phi_{13}}$. Finally, the following exponentiation is performed to accomplish the final exponentiation:

$$u_1 \leftarrow v_2^{x^3}, u_2 \leftarrow u_1^{x^3}, u_3 \leftarrow u_2^{x^3}, \tag{12}$$

$$u_4 \leftarrow v_2^{p^{10}} \cdot u_1^{p^7} \cdot u_2^{p^4} \cdot (u_3)^p \cdot v_1 \cdot (f^p)^3. \tag{13}$$

7 Operation Counts and Implementation Results

7.1 Operation counts

At the modified Miller loop, we require to compute Nf and Df, which means that two modified Miller iterations are executed. Denote N_1(resp. N_2) and D_1(resp. D_2) to the numerator and denominator of $f_{x,Q}(P)$(resp. $f_{x,[x]Q}(P)$), respectively. The binary representation the loop parameter x is 100010110000. At the first modified Miller iteration, we compute N_1 and D_1 by executing 3 quadrupling, 2 doubling and 3 doubling-addition steps, and $2\tilde{m} + \tilde{s} + 13m + \tilde{a}$ to recover $f_{x,Q}(P)$ from $g_{x,Q}(P)$. The values of N_2 and D_2 can be obtained by the same way, besides that we require to transform the point $[x]Q$ from Jacobian coordinates into affine coordinates at the beginning of the second modified Miller loop, which comes a cost of $\tilde{i} + 3\tilde{m} + \tilde{s}$. In order to compute $\ell_{\pi^2(Q),\pi([x]Q)}(P)$, one requires to compute $\pi^2(Q)$ and $\pi([x]Q)$ firstly, which costs $4f$. Denote $[x]Q = (x_1, y_1)$ and $\pi([x]Q) = (x_2, y_2)$ in affine coordinates. Then,

$$\begin{aligned}&\ell_{\pi^2(Q),\pi([x]Q)}(P)\\&= \frac{(y_P - y_1)\cdot(x_2 - x_1) - (y_2 - y_1)\cdot(x_P - x_1)}{x_2 - x_1}.\end{aligned}$$

Denote h by the numerator of $\ell_{\pi^2(Q),\pi([x]Q)}(P)$. The computation of h can also use the technique of lazy reduction, which requires $2\tilde{m}_u + 1\tilde{r} + 4f + 6\tilde{a}$. In total, we have

$$Nf = (N_1 \cdot D_1^{-1})^{p+x} \cdot N_2 \cdot h, \tag{14}$$

$$Df = D_2 \cdot (x_2 - x_1). \tag{15}$$

As proposed in [19], raising $N_1 \cdot D_1^{-1}$ to power of x only requires $4\tilde{m}$, since the squares can be shared with the computation of N_2. Hence, it costs $\tilde{i} + 8\tilde{m} + f$ in (14)-(15). We delay the computation of Nf/Df into the final exponentiation.

In conclusion, the total number of operations required in the Miller loop using Jacobian coordinates is

$$\begin{aligned}ML =&6(9\tilde{m} + 4\tilde{m}_u + 11\tilde{s} + 2\tilde{s}_u + 13m + 13m_u + 4\tilde{r} + 21\tilde{a})\\&+4(6\tilde{m} + 2\tilde{m}_u + 6\tilde{s} + \tilde{s}_u + 13m + 13m_u + 2\tilde{r} + 11\tilde{a})\\&+6(11\tilde{m} + 6\tilde{m}_u + 8\tilde{s} + \tilde{s}_u + 26m + 4\tilde{r} + 21\tilde{a})\\&+2(2\tilde{m} + \tilde{s} + 13m + \tilde{a}) + (\tilde{i} + 3\tilde{m} + \tilde{s})\\&+(2\tilde{m}_u + 1\tilde{r} + 4f + 6\tilde{a}) + (\tilde{i} + 8\tilde{m} + f)\\=&2\tilde{i} + 159\tilde{m} + 70\tilde{m}_u + 141\tilde{s} + 22\tilde{s}_u + 312m + 130m_u\\&+ 57\tilde{r} + 304\tilde{a} + 5f\\=&2i+518m+15798m_u+10758s_u+4747r+195187a.\end{aligned}$$

In the final exponentiation, computing $f = (Nf/Df)^{p-1}$ requires $\tilde{i} + 3\tilde{m} + 2f$. Raising f to the powers λ_0 and $\lambda_{10} + \lambda_{11}\cdot p + \lambda_{12}\cdot p^2$ totally cost $16\tilde{m} + 4\tilde{s} + 2\tilde{i}_c + 18e + 4f$. The last step of the final exponentiation as shown in (12)-(13) requires $6\tilde{m} + \tilde{s} + 9e + 5f$. Hence, the total number of operations required in the final exponentiation is

$$FE = (\tilde{i} + 3\tilde{m} + 2f) + (16\tilde{m} + 4\tilde{s} + 2\tilde{i}_c + 18e + 4f)$$

$$
\begin{aligned}
&+ (6\tilde{m} + \tilde{s} + 9e + 5f) \\
&= \tilde{i} + 2\tilde{i}_c + 25\tilde{m} + 5\tilde{s} + 27e + 11f \\
&= i + 325m + 7801m_u + 19932s_u + 193048a + 5461r.
\end{aligned}
$$

7.2 Implementation results

According to algorithms proposed in Section 3, we first implement finite field by using the C programming language. The codes are complied with the GCC compiler using -03 optimization level. Lower-level prime field operations are based on Version 0.5.0 of RELIC library [33]. Timing benchmarks are taken on a 64-bit Intel Core i7-8550U @1.8GHz processor running Ubuntu 18.04.1. We present the corresponding

Table 4. Operation counts averaged over 10^4 trials for finite field arithmetic

$\mathbb{F}_{p^{13}}$ Arithmetic	Operation Counts in $\mathbb{F}_p$	Clock Cycles
Addition	$13a$	566
Multiplication	$66m_u + 502a + 13r$	8788
Squaring	$66s_u + 443a + 13r$	3842
Inversion	$277m_u + 73m + i + 2034a + 53r$	39875
Inversion in $\mathbb{G}_{\phi_{13}}$	$264m_u + 60m + 2008a + 52r$	34211
Exponentiation by x	$198m_u + 726s_u + 6379a + 182r$	92619
Frobenius	$12m$	1904

Table 5. Comparison of implementations averaged over 10^4 trials between the *BW13-P310* curve and the*BLS12-446* curve

Curve	Phase	Clock Cycles
BLS12-446 [33]	Miller loop	1215923
	Final exponentiation	1227885
	Optimal ate pairing	2443808
BW13-P310	Miller loop	3597243
	Final exponentiation	2779631
	Optimal ate pairing	6376874

operation counts and timing benchmarks in Table 4. On this basis, we then implement the optimal ate pairing on the *BW13-P310* curve. Comparison of timing benchmarks between the *BW13-P310* and the *BLS12-446* curve are given in Table 5. The result shows that the implementations of pairings on the *BW13-P310* curve is about 2 times slower than that on the *BLS12-446* curve.

8 Conclusions

In this paper, we studied pairing computation on pairing friendly curves with odd prime embedding degrees, instantiated by using the *BW13-P310* curve on a 64-bit PC processor. The curve can reach 128-bit security level under the attack of SexTNFS. We proposed a modified Miller function and gave a new strategy to implement Miller's algorithm. Based on that, we drew a comparison between Jacobian and homogeneous coordinates.

In addition, we also presented an efficient method to execute final exponentiation. Even though pairing computation on the *BW13-P310* curve is slower than that on the *BLS12-446* curve, it is still meaningful since the size of prime field of the curve is smaller. As proposed by Clarisse, Duquesne and Sanders [18], the curve can be used in the schemes that requires an amount of exponentiation in $\mathbb{G}_1$, such as EPID and DAA.

Acknowledgement

This work is supported by Guangdong Major Project of Basic and Applied Basic Research(No. 2019B030302008), the National Natural Science Foundation of China(No.s 61972429 and 61972428) ,the Natural Science Foundation of Hunan Province of China (No.2021JJ40701) and the Research Fund of National University of Defense Technology (No. ZK20-42).

References

1. D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," Advances in Cryptology — CRYPTO 2001, ed. J. Kilian, Berlin, Heidelberg, pp.213–229, Springer Berlin Heidelberg, 2001.
2. D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," Advances in Cryptology — ASIACRYPT 2001, ed. C. Boyd, Berlin, Heidelberg, pp.514–532, Springer Berlin Heidelberg, 2001.
3. E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," Proceedings of the 11th ACM Conference on Computer and Communications Security, New York, NY, USA, pp.132–145, Association for Computing Machinery, 2004.
4. E. Brickell and J. Li, "Enhanced privacy id: A direct anonymous attestation scheme with enhanced revocation capabilities," IEEE Transactions on Dependable and Secure Computing, vol.9, no.3, pp.345–360, 2012.
5. N. El Mrabet and M. Joye, Guide to pairing-based cryptography, Chapman and Hall/CRC, 2016.
6. F. Hess, N.P. Smart, and F. Vercauteren, "The eta pairing revisited," IEEE Transactions on Information Theory, vol.52, no.10, pp.4595–4602, 2006.
7. S. Matsuda, N. Kanayama, F. Hess, and E. Okamoto, "Optimised versions of the ate and twisted ate pairings," Cryptography and Coding, ed. S.D. Galbraith, Berlin, Heidelberg, pp.302–312, Springer Berlin Heidelberg, 2007.
8. C.A. Zhao, F. Zhang, and J. Huang, "A note on the ate pairing," International Journal of Information Security, vol.7, no.6, pp.379–382, 2008.
9. E. Lee, H.S. Lee, and C.M. Park, "Efficient and generalized pairing computation on abelian varieties," IEEE Transactions on Information Theory, vol.55, no.4, pp.1793–1803, 2009.

10. C.A. Zhao, F. Zhang, and J. Huang, "All pairings are in a group," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.91, no.10, pp.3084–3087, 2008.
11. F. Vercauteren, "Optimal pairings," IEEE Transactions on Information Theory, vol.56, no.1, pp.455–461, 2009.
12. J.M. Pollard, "A monte carlo method for factorization," Bit Numerical Mathematics, vol.15, no.3, pp.331–334, 1975.
13. T. Kim and R. Barbulescu, "Extended tower number field sieve: A new complexity for the medium prime case," Advances in Cryptology – CRYPTO 2016, ed. M. Robshaw and J. Katz, Berlin, Heidelberg, pp.543–571, Springer Berlin Heidelberg, 2016.
14. Gordon and M. Daniel, "Discrete logarithms in GF(p) using the number field sieve," SIAM Journal on Discrete Mathematics, vol.6, no.1, pp.124–138, 1993.
15. O. Schirokauer, "Discrete logarithms and local units," Philosophical Transactions: Physical Sciences and Engineering, vol.345, no.1676, pp.409–423, 1993.
16. O. Schirokauer, "Using number fields to compute logarithms in finite fields," Mathematics of Computation, vol.69, pp.1267–1283, 2000.
17. A. Joux, R. Lercier, N. Smart, and F. Vercauteren, "The number field sieve in the medium prime case," Advances in Cryptology - CRYPTO 2006, ed. C. Dwork, Berlin, Heidelberg, pp.326–344, Springer Berlin Heidelberg, 2006.
18. R. Clarisse, S. Duquesne, and O. Sanders, "Curves with fast computations in the first pairing group," Cryptology and Network Security, ed. S. Krenn, H. Shulman, and S. Vaudenay, Cham, pp.280–298, Springer International Publishing, 2020.
19. A. Guillevic, "A short-list of pairing-friendly curves resistant to special tnfs at the 128-bit security level," Public-Key Cryptography – PKC 2020, ed. A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, Cham, pp.535–564, Springer International Publishing, 2020.
20. A. Guillevic, S. Masson, and E. Thomé, "Cocks-pinch curves of embedding degrees five to eight and optimal ate pairing computation," Designs Codes and Cryptography, vol.88, no.4, pp.1047–1081, 2020.
21. D. Freeman, M. Scott, and E. Teske, "A taxonomy of pairing-friendly elliptic curves," Journal of cryptology, vol.23, no.2, pp.224–280, 2010.
22. W. Bosma, J. Cannon, and C. Playoust, "The Magma algebra system. I. The user language," J. Symbolic Comput., vol.24, no.3-4, pp.235–265, 1997.
23. D.F. Aranha, K. Karabina, P. Longa, C.H. Gebotys, and J. López, "Faster explicit formulas for computing pairings over ordinary curves," Advances in Cryptology – EUROCRYPT 2011, ed. K.G. Paterson, Berlin, Heidelberg, pp.48–68, Springer Berlin Heidelberg, 2011.
24. X. Lin, C.A. Zhao, F. Zhang, and Y. Wang, "Computing the ate pairing on elliptic curves with embedding degree k=9," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E91.A, no.9, pp.2387–2393, 2008.
25. Y. Nanjo, M. Shirase, T. Kusaka, and Y. Nogami, "Improvement of final exponentiation for pairings on bls curves with embedding degree 15," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E104.A, no.1, pp.315–318, 2021.
26. L. Fuentes-Castañeda, E. Knapp, and F. Rodríguez-Henríquez, "Faster hashing to $\mathbb{G}_2$," Selected Areas in Cryptography, ed. A. Miri and S. Vaudenay, Berlin, Heidelberg, pp.412–430, Springer Berlin Heidelberg, 2012.
27. R. Lidl and H. Niederreiter, Introduction to finite fields and their applications, Cambridge university press, 1994.
28. A.J. Devegili, M. Scott, and R. Dahab, "Implementing cryptographic pairings over barreto-naehrig curves," Pairing-Based Cryptography – Pairing 2007, ed. T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, Berlin, Heidelberg, pp.197–207, Springer Berlin Heidelberg, 2007.

29. A. Weimerskirch and C. Paar, "Generalizations of the karatsuba algorithm for efficient implementations," IACR Cryptol. ePrint Arch., vol.2006, p.224, 2006.
30. "Explicit-formulas database: Genus-1 curves over large-characteristic fields." https://www.hyperelliptic.org/EFD/g1p/auto-shortw.html.
31. R. Azarderakhsh, D. Fishbein, G. Grewal, S. Hu, D. Jao, P. Longa, and R. Verma, "Fast software implementations of bilinear pairings," IEEE Transactions on Dependable and Secure Computing, vol.14, no.6, pp.605–619, 2017.
32. C. Costello, C. Boyd, J.M. González Nieto, and K.K.H. Wong, "Avoiding full extension field arithmetic in pairing computations," Progress in Cryptology – AFRICACRYPT 2010, ed. D.J. Bernstein and T. Lange, Berlin, Heidelberg, pp.203–224, Springer Berlin Heidelberg, 2010.
33. D.F. Aranha and C.P.L. Gouvêa, "RELIC is an Efficient LIbrary for Cryptography." https://github.com/relic-toolkit/relic.

A Sub-Algorithm

Algorithm 3: $MulLevel2(a, b)$ cost $=3m_u + 6a$

Input: $a = a_0 + a_1\omega, b = b_0 + b_1\omega$
Output: $u = a \times b$

$u_0 \leftarrow a_0 \times b_0, u_1 \leftarrow a_1 \times b_1$
$t_0 \leftarrow a_0 + b_0, t_1 \leftarrow a_1 + b_1$
$u_2 \leftarrow t_0 \times t_1, u_2 \leftarrow u_2 - u_0$
$u_2 \leftarrow u_2 - u_1$
return
$u = u_0 + u_1\omega + u_2\omega^2$

Algorithm 4: $MulLevel3(a, b)$ cost=$6m_u$+$20a$

Input: $a = \sum_{i=0}^{2} a_i\omega^i, b = \sum_{i=0}^{2} b_i\omega^i$
Output: $u = a \times b$

1: $u_0 \leftarrow a_0 \times b_0$
2: $u_2 \leftarrow a_1 \times b_1$
3: $u_4 \leftarrow a_2 \times b_2$
4: $t_0 \leftarrow a_0 + a_1$
5: $t_1 \leftarrow b_0 + b_1$
6: $u_1 \leftarrow t_0 \times t_1$
7: $u_1 \leftarrow u_1 - u_0$
8: $u_1 \leftarrow u_1 - u_2$
9: $t_0 \leftarrow a_1 + a_2$
10: $t_1 \leftarrow b_1 + b_2$
11: $u_3 \leftarrow t_0 \times t_1$
12: $u_3 \leftarrow u_3 - u_2$
13: $u_3 \leftarrow u_3 - u_4$
14: $t_0 \leftarrow a_0 + a_2$
15: $t_1 \leftarrow b_0 + b_2$
16: $\tilde{u} \leftarrow t_0 \times t_1$
17: $\tilde{u} \leftarrow \tilde{u} - u_0$
18: $\tilde{u} \leftarrow \tilde{u} - u_4$
19: $u_2 \leftarrow u_2 + \tilde{u}$
20: **return**
$u = \sum_{i=0}^{4} u_i\omega^i$

Algorithm 5: $MulLevel4(a, b)$ cost=$9m_u$+$38a$

Input: $a = \sum_{i=0}^{3} a_i\omega^i, b = \sum_{i=0}^{3} b_i\omega^i$

Output: $u = a \times b$

$A_0 \leftarrow a_0 + a_1\omega$
$A_1 \leftarrow a_2 + a_3\omega$
$B_0 \leftarrow b_0 + b_1\omega$
$B_1 \leftarrow b_2 + b_3\omega$
$\alpha \leftarrow MulLevel2(A_0, B_0)$
$\beta \leftarrow MulLevel2(A_1, B_1)$
$T_0 \leftarrow A_0 + A_1$
$T_1 \leftarrow B_0 + B_1$
$\gamma \leftarrow MulLevel2(T_0, T_1)$
for $i = 0$ **to** 2 **do**
    $\gamma_i \leftarrow \gamma_i - \alpha_i$
    $\gamma_i \leftarrow \gamma_i - \beta_i$
end for
$u_0 \leftarrow \alpha_0$,
$u_1 \leftarrow \alpha_1$
$u_2 \leftarrow \alpha_2 + \gamma_0$
$u_3 \leftarrow \gamma_1$
$u_4 \leftarrow \beta_0 + \gamma_2$
$u_5 \leftarrow \beta_1$,
$u_6 \leftarrow \beta_2$
return $u = \sum_{i=0}^{6} u_i\omega^i$

Algorithm 6: $MulLevel6(a, b)$ cost=$18m_u$+$94a$

Input: $a = \sum_{i=0}^{5} a_i\omega^i, b = \sum_{i=0}^{5} b_i\omega^i$

Output: $u = a \times b$

$A_0 \leftarrow a_0 + a_1\omega + a_2\omega^2$
$A_1 \leftarrow a_3 + a_4\omega + a_5\omega^2$
$B_0 \leftarrow b_0 + b_1\omega + b_2\omega^2$
$B_1 \leftarrow b_3 + b_4\omega + b_5\omega^2$
$\alpha \leftarrow MulLevel3(A_0, B_0)$
$\beta \leftarrow MulLevel3(A_1, B_1)$
$T_0 \leftarrow A_0 + A_1, T_1 \leftarrow B_0 + B_1$
$\gamma \leftarrow MulLevel3(T_0, T_1)$
for $i = 0$ **to** 4 **do**
    $u_i \leftarrow \alpha_i + \beta_i$
    $\gamma_i \leftarrow \gamma_i - u_i$
end for
$u_0 \leftarrow \alpha_0$,
$u_1 \leftarrow \alpha_1$
$u_2 \leftarrow \alpha_2$,
$u_3 \leftarrow \alpha_3 + \gamma_0$
$u_4 \leftarrow \alpha_4 + \gamma_1$
$u_5 \leftarrow \gamma_2$
$u_6 \leftarrow \beta_0 + \gamma_3$
$u_7 \leftarrow \beta_1 + \gamma_4$
$u_8 \leftarrow \beta_2$
$u_9 \leftarrow \beta_3$
$u_{10} \leftarrow \beta_4$
return $u = \sum_{i=0}^{10} u_i\omega^i$

Algorithm 7: $MulLevel7(a,b)$ cost=$24m_u$+$138a$

Input: $a=\sum_{i=0}^{6} a_i\omega^i, b=\sum_{i=0}^{6} b_i\omega^i$

Output: $u=a\times b$

$A_0 \leftarrow a_0+a_1\omega+a_2\omega^2$
$A_1 \leftarrow a_3+a_4\omega+a_5\omega^2+a_6\omega^3$
$B_0 \leftarrow b_0+b_1\omega+b_2\omega^2$
$B_1 \leftarrow b_3+b_4\omega+b_5\omega^2+b_6\omega^3$
$\alpha \leftarrow MulLevel3(A_0,B_0)$
$\beta \leftarrow MulLevel4(A_1,B_1)$
$T_0 \leftarrow A_0+A_1\ T_1 \leftarrow B_0+B_1$
$\gamma \leftarrow MulLevel4(T_0,T_1)$
for $i=0$ **to** 4 **do**
    $u_i \leftarrow \alpha_i+\beta_i$
    $\gamma_i \leftarrow \gamma_i-u_i$
end for
$\gamma_5 \leftarrow \gamma_5-\beta_5$
$\gamma_6 \leftarrow \gamma_6-\beta_6$
$u_0 \leftarrow \alpha_0$
$u_1 \leftarrow \alpha_1, u_2 \leftarrow \alpha_2$
$u_3 \leftarrow \alpha_3+\gamma_0$
$u_4 \leftarrow \alpha_4+\gamma_1$
$u_6 \leftarrow \beta_0+\gamma_3$
$u_7 \leftarrow \beta_1+\gamma_4$
$u_8 \leftarrow \beta_2+\gamma_5$
$u_9 \leftarrow \beta_3+\gamma_6$
$u_5 \leftarrow \gamma_2, u_8 \leftarrow \beta_2$
$u_9 \leftarrow \beta_3, u_{10} \leftarrow \beta_4$
$u_{11} \leftarrow \beta_5$
$u_{12} \leftarrow \beta_6$
return $u=\sum_{i=0}^{12} u_i\omega^i$