Revisiting subgroup membership testing on pairing-friendly curves via the Tate pairing

Yu Dai[1], Debiao He[2✉], Dmitrii Koshelev[3], Cong Peng[2], and Zhijian Yang[1]

[1] School of Mathematics and statistics, Wuhan University, Wuhan, China.
eccdaiy39@gmail.com, zjyang.math@whu.edu.cn

[2] School of Cyber Science and Engineering, Wuhan University, Wuhan, China.
hedebiao@whu.edu.cn, cpeng@whu.edu.cn

[3] Department of Mathematics, University of Lleida, Catalonia, Spain.
dimitri.koshelev@gmail.com

Abstract. In 2023, Koshelev proposed an efficient method for subgroup membership testing on a list of non-pairing-friendly curves via the Tate pairing. In fact, this method can also be applied to certain pairing-friendly curves, such as the BLS and BW13 families, at a cost of two small Tate pairings. In this paper, we revisit Koshelev's method to enhance its efficiency for these curve families. First, we present explicit formulas for computing the two small Tate pairings. Compared to the original formulas, the new versions offer shorter Miller iterations and reduced storage requirements. Second, we provide a high-speed software implementation on a 64-bit processor. Our results demonstrate that the new method is up to 62.0% and 22.4% faster than the state-of-the-art on the BW13-310 and BLS24-315 curves, respectively, while being 14.1% slower on BLS12-381. When precomputation is utilized, our method achieves speed improvements of up to 34.8%, 110.6%, and 63.9% on the BLS12-381, BW13-310, and BLS24-315 curves, respectively.

Keywords: pairing-friendly curves· subgroup membership testing· Tate pairing.

1 Introduction

A cryptographic pairing on an elliptic curve E over a prime field $\mathbb{F}_p$ is a bilinear and non-degenerate map of the form $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three subgroups with the same large prime order r. More specifically, the two input groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are distinct subgroups of $E(\mathbb{F}_{p^k})$ and the output group $\mathbb{G}_T$ is subgroup of a finite field $\mathbb{F}_{p^k}$, where k is the smallest positive integer such that $r \mid (p^k - 1)$. Over the last two decades, pairings have been widely used in the design of various cryptographic protocols. Nowadays, the research on pairings remains active, driven largely by their applications in Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs), such as Groth16 [23] and PlonK [20]. In pairing-based cryptographic protocols, participants often need to perform exponentiation operations in one or more pairing groups. However, these groups usually lie in a larger group with non-trivial cofactors for most of mainstream pairing-friendly curves. As a result, this can lead to small subgroup attacks: if a participant performs group exponentiationon an element of non-prime order with a secret key, it may expose partial information about the secret key. We refer the readers to [6] for details.

In order to resist small subgroup attacks in pairing-based cryptographic protocols, it is essential to ensure that a given element belongs to a specific subgroup. This process is called *subgroup membership testing*. Given a candidate element g claimed to be a member of $\mathbb{G}_i$ $(i \in \{1, 2, T\})$, the naive method involves checking whether g^r is equal to the identity element of $\mathbb{G}_i$. However, the cost of this method is expensive as the size of the prime r is large in practice. In 2020, Scott [41] proposed an efficient method of subgroup membership testings on the BLS12 curves via efficiently computable endmoriphisms. This method is about $2\times$, $4\times$ and $4\times$ as fast as the naive one for $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ membership testings, respectively. Subsequently, Dai et al. [13] generalized Scott's method such that it can be applied to a large class of pairing-friendly curves. In more detail, Dai et al.'s method requires about $\log r/2$, $\log r/\varphi(k)$ and $\log r/\varphi(k)$ group operations for $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ membership testings respectively, where $\varphi(k)$ is the Euler phi function of k. In particular, the number of group operations for $\mathbb{G}_2$ can be further reduced to around $\log r/(2\varphi(k))$ on some special pairing-friendly curves [12]. It should be also noted that the cost of $\mathbb{G}_2$ membership testing might be free if it is allowed to execute the testing during pairing computation.

Recently, Koshelev [32] proposed a novel method for subgroup membership testing on non-pairing-friendly curves via the Tate pairing. This method imposes specific requirements for the group structure of elliptic curves. To be precise, given an elliptic curve E over a finite field $\mathbb{F}_p$ with a large prime divisor r, it follows from [43, Theorem 4.1] that $E(\mathbb{F}_p) \cong \mathbb{Z}_{e_1} \oplus \mathbb{Z}_{e_2 \cdot r}$ for some integers e_1 and e_2 with $e_1 \mid e_2$. Koshelev stated that if $e_2 \mid (p-1)$ one can perform subgroup membership testing for $E(\mathbb{F}_p)[r]$ via two Tate pairings of orders e_1 and e_2, respectively. In particular, if $E(\mathbb{F}_p)$ is cyclic, i.e., $e_1 = 1$, it only requires computing one e_2-order pairing. For example, the given method is well-suited for Jubjub [18] and Bandersnatch [34]. Since $e_1 \in \{1, 2\}$ and $e_2 \in \{2, 8\}$ on these curves, the length of the Miller loop is extremely short, and thus the whole pairing computation amounts only to its final exponentiation. Curiously, the latter can be also performed effciently when $e_2 \leq 11$ according to [29]. It should be noted that Koshelev [33] subsequently generalized his method such that it can be applied to certain curves with $e_2 \nmid (p-1)$, e.g., Ed448-Goldilocks [26]. Unfortunately, this generalization may be inefficient as it involves operations in an extension field $\mathbb{F}_{p^d}$, where d is the smallest degree such that the exponent of the group $E(\mathbb{F}_{p^d})[e_2^\infty]$ divides $p^d - 1$.

In fact, Koshelev's method is also suitable for $\mathbb{G}_1$ membership testing on a list of mainstream pairing-friendly curves, such as the Barreto-Lynn-Scott(BLS) [8] family and complete families from [19, Construction 6.6] with embedding degrees 13 and 19. The latter is also referred to as BW curves in the literature, as they are constructed via the Brezing-Weng method [9]. However, when applying

Koshelev's method to these curves, the computational cost of the Miller loop cannot be ignored. For example, the sizes of e_1 and e_2 are both approximately 1/4 that of the prime r for the BLS12 curves, making Koshelev's method more computationally expensive than Scott's method.

1.1 Our contributions

The goal of this work is to illustrate how to efficiently apply Koshelev's method to accelerate $\mathbb{G}_1$ membership testing in the BLS, BW13 and BW19 families. Our contributions are summarized as follows.

- We present an efficient algorithm for computing the two Tate pairings required for subgroup membership testing on our target curves. The new algorithm requires approximately $\log e_2$ Miller's iterations. In addition, we also demonstrate how to perform the the final exponentiation of the e_1-order Tate pairing.
- Using the RELIC cryptographic toolkit [1], we present a high-speed software implementation across a list of mainstream curves, including BLS12-381, BLS12446, BW13-310, BLS24-315, BLS24-509 and BLS48-575. The experimental results show that
    1. in the general case, our method achieves speed improvements of up to 62.0%, 22.4%, 46.2% and 80.6% on the BW13-310, BLS24-315, BLS24-509 and BLS48-575 curves, respectively, while being approximately 14.6% and 41.4% slower on BLS12-381 and BLS12-446 curves, respectively;
    2. when precomputation is utilized, our method is approximately 34.8%, 10.6%, 110.6%, 63.9%, 98.1% and 123.1% faster than the previous leading work on the BLS12-381, BLS12-446, BW13-310, BLS24-315, BLS24-509 and BLS48-575 curves, respectively.

Organization of the Paper. The remainder of this paper is organized as follows. In Section 2, we give some preliminaries about elliptic curves used in this work. Section 3 introduces existing methods for subgroup membership testing on elliptic curves. In Section 4, we present new formulas for subgroup membership testings that suitable for the BLS, BW13 and BW19 families. Section 5 offers a comprehensive performance comparison between our implementation and the previous fastest one. Finally, we draw our conclusion in Section 6.

2 Preliminaries

In this section, we first review some basic concepts about elliptic curves and Tate pairings. Then we introduce some parameterized pairing-friendly curves.

2.1 Ordinary elliptic curves

Let $\mathbb{F}_p$ be a prime field with $p > 3$ a large prime. An elliptic curve E over $\mathbb{F}_p$ in the short Weierstrass form is defined by the equation $y^2 = x^3 + ax + b$, where $a, b \in \mathbb{F}_p$ such that $4a^3 + 27b^2 \neq 0$. The j-invariant of E is given by $j(E) = 1728\frac{4a^3}{4a^3+27b^2}$. The group $E(\mathbb{F}_p)$ consists of $\mathbb{F}_p$-rational points (x, y) that satisfy the curve equation, together with the identity point $\mathcal{O}$. The order of $E(\mathbb{F}_p)$, which is denoted as $\#E(\mathbb{F}_p)$, is equal to $p + 1 - t$, where t is the trace of the p-power Frobenius map $\pi : (x, y) \to (x^p, y^p)$. If $t = 0$, then the curve E is said to be *supersingular*; otherwise it is *ordinary*. Let r be a large prime such that $r \parallel \#E(\mathbb{F}_p)$, and let $E[r]$ be the r-torsion group of E. The embedding degree k of E with respect to r and p is the smallest positive integer such that $E[r] \subseteq E(\mathbb{F}_{p^k})$. If $k > 1$, then k is the smallest integer such that $r|(p^k - 1)$. In this case, the group $E[r] = \mathbb{G}_1 \oplus \mathbb{G}_2$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are defined as follows:

$$\mathbb{G}_1 = E(\mathbb{F}_p)[r] \text{ and } \mathbb{G}_2 = E(\mathbb{F}_{p^k})[r] \cap \mathrm{Ker}(\pi - [p]).$$

The group $E(\mathbb{F}_p) \cong \mathbb{Z}_{e_1} \oplus \mathbb{Z}_{e_2 \cdot r}$ for some integers e_1 and e_2 with $e_1 \mid e_2$. If E is an ordinary curve with j-invariant 0 or 1728, then the cofactor e_1 is the largest integer such that $e_1^2 \mid \#E(\mathbb{F}_p)$ and $e_1 \mid (p - 1)$ [39, Proposition 3.7]. As a result, the cofactor $e_2 = me_1$, where $m = \#E(\mathbb{F}_p)/(r \cdot e_1^2)$. For any integer n satisfying that $n \mid (p - 1)$ and $n \mid \#E(\mathbb{F}_p)$, i.e., n is a divisor of e_2, the reduced Tate pairing of order n on $E(\mathbb{F}_p)$ is defined as

$$\begin{aligned} T_n : E(\mathbb{F}_p)[n] \times E(\mathbb{F}_p)/nE(\mathbb{F}_p) &\to \mu_n \\ (P, R) &\to f_{n,P}(R)^{(p-1)/n}, \end{aligned}$$

where $\mu_n \subseteq \mathbb{F}_p^*$ is the group of n-th roots of unity in $\mathbb{F}_p^*$ and $f_{n,P}$ is a rational function with divisor $\mathrm{div}(f_{n,P}) = (n)P - (nP) - (n - 1)\mathcal{O}$. The Tate pairing T_n on E satisfies the following properties:

1. **Bilinearity**: For any $P_1, P_2 \in E(\mathbb{F}_p)[n]$ and $R_1, R_2 \in E(\mathbb{F}_p)$,

$$\begin{aligned} T_n(P_1, R_1 + R_2) &= T_n(P_1, R_1) \cdot T_n(P_1, R_2), \\ T_n(P_1 + P_2, R_1) &= T_n(P_1, R_1) \cdot T_n(P_2, R_1). \end{aligned}$$

2. **Non-degeneracy**: Let $P \in E(\mathbb{F}_p)[n]$. If $T_n(P, R) = 1$ for all $R \in E(\mathbb{F}_p)$, then $P = \mathcal{O}$. Let $R \in E(\mathbb{F}_p)$. If $T_n(P, R) = 1$ for all $P \in E(\mathbb{F}_p)[n]$, then $R \in nE(\mathbb{F}_p)$.
3. **Compatibility**: Let $P \in E(\mathbb{F}_p)[n]$ and $R \in E(\mathbb{F}_p)$. For all separable endomorphisms α on E,

$$T_n(\alpha(P), R) = T_n(P, \hat{\alpha}(R)),$$

where $\hat{\alpha}$ is the dual of α, i.e., $\alpha \circ \hat{\alpha} = [\deg(\alpha)]$.

Throughout this paper, we assume the curve E is ordinary with j-invariant 0. In this case, the parameter $a = 0$ and $p \equiv 1 \bmod 3$ [43, Proposition 4.33]. Therefore, there exists an efficiently computable endomorphism ϕ on E:

$$\begin{aligned} \phi : &E \to E \\ &(x, y) \to (\omega \cdot x, y), \end{aligned}$$

Algorithm 1 Miller's Algorithm

```
Input: $P \in E(\mathbb{F}_p)[n]$, $R \in E(\mathbb{F}_p)$, $m = \sum_{i=0}^{l} m_i 2^i$ with $m_i \in \{0,1\}$
Output: $f_{m,P}(R)$
1: $T \leftarrow P$, $f \leftarrow 1$
2: for $i = l-1$ down to $0$ do
3:     $f \leftarrow f^2 \cdot \frac{\ell_{T,T}(R)}{\nu_{2T}(R)}$, $T \longleftarrow 2T$
4:     if $m_i = 1$ then
5:         $f \leftarrow f \cdot \frac{\ell_{T,P}(R)}{\nu_{T+P}(R)}$, $T \leftarrow T + P$
6:     end if
7: end for
8: return $f$
```

where ω is a primitive cube root of unity in $\mathbb{F}_p^*$. The dual of ϕ is given as $\hat{\phi} : (x, y) \to (\omega^2 \cdot x, y)$. By the property of compatibility of Tate pairing, we have

$$T_n(\phi(P), R) = T_n(P, \hat{\phi}(R)).$$

The endomorphism ϕ is also called the GLV endomorphism as it was used by Gallant, Lambert and Vanstone [22] to accelerate elliptic curve scalar multiplication.

The computation of the Tate pairing $T_n(P, R)$ can be divided into two phases: the Miller loop and the final exponentiation. In more detail, the first phase involves computing $f_{n,P}(R)$ and the second one aims to raise $f_{n,P}(R)$ to the power of $(p-1)/n$. The computation of $f_{n,P}(R)$ can be executed via Miller's algorithm [36], which is actually based on the following equation:

$$f_{i+j,P} = f_{i,P} \cdot f_{j,P} \cdot \frac{\ell_{iP,jP}}{\nu_{(i+j)P}}, \qquad (1)$$

where $\ell_{iP,jP}$ represents the straight line through iP and jP, and $\nu_{(i+j)P}$ denotes the vertical line through $(i+j)P$. In the following, we refer to $f_{n,P}$ as the Miller function.

2.2 Parameterized families of pairing-friendly curves

A cryptographic pairing is a bilinear and non-degenerate map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$, where $\mathbb{G}_T$ represents the r-th roots of unity in $\mathbb{F}_{p^k}^*$. Pairing-friendly curves are specifically designed to facilitate high-performance implementations of pairing computations at the required security levels. These curves typically have a low embedding degree k and a small value of ρ, where $\rho = \log p / \log r$. Table 1 summarizes some popular parameterized families of pairing-friendly curves with $\rho < 2$ and embedding degree k ranging from 12 to 48, including the BW13, BW19, BLS12, BLS24, and BLS48 families. All these curves have a j-invariant 0, meaning they can be defined by the equation $y^2 = x^3 + b$ for some $b \in \mathbb{F}_p^*$.

Table 1: Polynomial parameters of BW13, BW19, BLS12, BLS24 and BLS48 families. The symbol $\Phi_k(z)$ represents the k-th cyclotomic polynomial.

family-k	p	r	t	e_1	e_2	m
BLS12	$(z-1)^2(z^4-z^2+1)/3+z$	$\Phi_{12}(z)$	$z+1$	$\frac{(z-1)}{3}$	$z-1$	3
BW13	$\frac{1}{3}(z+1)^2(z^{26}-z^{13}+1)-z^{27}$	$\Phi_{78}(z)$	$-z^{14}+z+1$	$\frac{(z^2-z+1)}{3}$	z^2-z+1	3
BW19	$\frac{1}{3}(z+1)^2(z^{38}-z^{19}+1)-z^{39}$	$\Phi_{114}(z)$	$-z^{20}+z+1$	$\frac{(z^2-z+1)}{3}$	z^2-z+1	3
BLS24	$(z-1)^2(z^8-z^4+1)/3+z$	$\Phi_{24}(z)$	$z+1$	$\frac{(z-1)}{3}$	$z-1$	3
BLS48	$(z-1)^2(z^{16}-z^8+1)/3+z$	$\Phi_{48}(z)$	$z+1$	$\frac{(z-1)}{3}$	$z-1$	3

Table 2: A list of pairing-friendly curves derived from the BLS, BW13 and BW19 families.

curve	b	z	$\lceil\log p\rceil$	$\lceil\log r\rceil$	ρ	$\lceil\log p^k\rceil$
BLS12-381	4	$-2^{63}-2^{62}-2^{60}-2^{57}-2^{48}-2^{16}$	381	255	1.5	4569
BLS12-446	1	$-2^{74}-2^{73}-2^{63}-2^{57}-2^{50}-2^{17}-1$	446	299	1.5	5376
BW13-310	-17	$-2^{11}-2^{7}-2^{5}-2^{4}$	310	267	1.17	4027
BW19-286	31	$-2^{7}-2^{4}-1$	286	259	1.11	5427
BLS24-315	1	$-2^{31}-2^{30}+2^{21}+2^{20}+1$	315	253	1.25	7543
BLS24-509	1	$-2^{51}-2^{28}+2^{11}-1$	509	409	1.25	12202
BLS48-575	4	$2^{32}-2^{18}-2^{10}-2^{4}$	575	512	1.125	27572

Due to the decrease of the asymptotic complexity for computing discrete logarithms in finite fields under the attacks of number field sieve and its variants [5,28,31], the parameters of pairing-friendly curves should be selected carefully to reach the desired security level. In Table 2, we list the key parameters of specific pairing-friendly curves derived from the above families, which are suitable for implementing pairing-based protocols across various security levels. To be precise, BLS12-381 is one of the most popular curves in practice, which is widely used for digital signatures and zero-knowledge proofs; BLS12-446, BLS24-509 and BLS48-575 are believed to be the best choice for pairing computations at the 128-bit, 192-bit and 256-bit security levels, respectively [2,24,35]; BW13-310 and BW19-286 provide fast performance of group exponentiation in $\mathbb{G}_1$ at the 128-bit security level [10]; BLS24-315 [16] is suitable for constructing zero-knowledge succinct non-interactive arguments of knowledge that based on

KZG [30] polynomial commitment, e.g., PLONK [20]. For more information on selecting pairing-friendly curves, we refer to Guillevic's blog [25].

3 Efficient Methods for Subgroup Membership Testing on Elliptic Curves

In this section, we recall the efficient methods for subgroup membership testing on elliptic curves in the existing literature.

3.1 Method I: subgroup membership testing via the efficiently computable endomorphism

Scott [41] proposed the first non-trivial method for $\mathbb{G}_1$ membership testing specifically designed for the BLS12 family. After that, EI Housni, Guillevic and Piellard [17] confirmed that this method was also suitable for the BLS24 and BLS48 families. In 2023, Dai et al. [13] further generalized Scott's method such that it can be applied to a large class of pairing-friendly curves. In essence, this method requires that pairing-friendly curves are equipped with an efficiently-computable endomorphism. In particular, for ordinary curves with j-invariant 0, Dai et al.'s method can be summarized as follows. Let the GLV endomorphism ϕ on $\mathbb{G}_1$ act as a scalar multiplication by λ_1, and define a two dimensional lattice $\mathcal{L}_\phi$ as

$$\mathcal{L}_\phi = \{(b_0, b_1) \in \mathbb{Z}^2 | b_0 + b_1 \cdot \lambda_1 \equiv 0 \bmod r\}.$$

We denote by $\mathbf{Res}(g_0, g_1)$ the resultant of the two polynomials g_0 and g_1. Let $\mathbf{a} = (a_0, a_1) \in \mathcal{L}_\phi$ such that

$$\gcd\left(\#E(\mathbb{F}_p), \mathbf{Res}(a_0 + a_1 X, X^2 + X + 1)\right) = r. \tag{2}$$

Dai et al. observed that

$$R \in \mathbb{G}_1 \Leftrightarrow R \in E(\mathbb{F}_p) \text{ and } a_0 R + a_1 \phi(R) = \mathcal{O}.$$

For efficiency, we expect that $\|\mathbf{a}\|_\infty$ is as small as possible. According to [42, Theorem 7], there exists a shortest vector $\mathbf{v}$ in $\mathcal{L}_\phi$ such that $\|\mathbf{v}\|_\infty \approx \sqrt{r}$. Fortunately, the condition (2) is generally mild, allowing the target short vector $\mathbf{a}$ can be always selected as $\mathbf{v}$ for many popular pairing-friendly curves. Consequently, this method requires approximately $\log r/2$ bit group operations, which is roughly twice as fast as the naive one.

3.2 Method II: subgroup membership testing via the Tate pairing

Taking advantage of the non-degeneracy of Tate pairing, Koshelev [32] proposed a new method for subgroup membership testing on elliptic curves. The essence of this method is captured in the following theorem.

Theorem 1. [32, **Lemma 1**] *Let E be an elliptic curve defined over $\mathbb{F}_p$ with $E(\mathbb{F}_p) \cong \mathbb{Z}_{e_1} \oplus \mathbb{Z}_{e_2 \cdot r}$ and $e_2 \mid (p-1)$. Let $P_1, P_2 \in E(\mathbb{F}_p)$ with orders e_1 and e_2 respectively, such that $E(\mathbb{F}_p)[e_2] = \langle P_1 \rangle \oplus \langle P_2 \rangle$. Given a random non-identity point $R \in E(\mathbb{F}_p)$, then $R \in E(\mathbb{F}_p)[r]$ if and only if*

$$T_{e_1}(P_1, R) = 1 \text{ and } T_{e_2}(P_2, R) = 1.$$

According to Theorem 1, subgroup membership testing on elliptic curves can be accomplished at a cost of two Tate pairings of orders e_1 and e_2, respectively. If $E(\mathbb{F}_p)$ is cyclic, i.e., $e_1 = 1$, it only requires computing the e_2-order Tate pairing. In addition, Koshelev also demonstrated that if $e_i \leq 11$ for $i \in \{1, 2\}$, then the final exponentiation part of the e_i-order Tate pairing can be sped up by using the Euclidean-type algorithm [29]. In particular, if $e_i = 2$, then the final exponentiation is equivalent to computing the Legendre symbol, which can be further accelerated using the algorithms presented in [3, 27].

3.3 Summary

The above two methods differ significantly in terms of applicability, efficiency and storage requirements. Specifically, Method I is tailored for pairing-friendly curves, while Method II was originally developed for non-pairing-friendly curves such as Jubjub [18] and Bandersnatch [34]. In summary, the main differences between the two methods are summarized as follows:

1. Method I is well-suited for elliptic curves equipped with efficiently computable endmoriphisms, while Method II is applicable when the curve parameters meet the condition $e_2 \mid (p-1)$.
2. The computational cost of Method I primarily arises from scalar multiplication with a bit length of approximately $\log r/2$. In contrast, the computational cost of Method II mainly comes from the two Tate pairing computations of orders e_1 and e_2, respectively.
3. Method I does not increase storage requirements since the parameter of the endomorphism ϕ is often provided to accelerate elliptic curve scalar multiplication in pairing-based systems. As a comparison, Method II requires storing two points P_1 and P_2 such that $E(\mathbb{F}_p)[e_2] = \langle P_1 \rangle \oplus \langle P_2 \rangle$.

It is clear that Method I is suitable for the BLS, BW13 and BW19 families. Fortunately, we notice that these families also meet the condition $e_2 \mid (p-1)$, making Method II applicable as well. However, despite the relatively small sizes of e_1 and e_2 in these families, the computational cost of the Miller loop remains non-negligible. In addition, the technique of [29] can not be exploited to speed up the final exponentiation as $e_i \gg 11$ for $i \in \{1, 2\}$. To the best of our knowledge, Method I is still considered the state-of-the-art of subgroup membership testing (for $\mathbb{G}_1$) on pairing-friendly curves. This method has been implemented in MIRACL [40] and RELIC [1].

4 Small Tate Pairing Computations in the BLS, BW13 and BW19 Families

In this section, we revisit Method II and study how to reduce its computational cost and storage requirements when applied to the BLS, BW13 and BW19 families.

Notations. We represent the points R and $\hat{\phi}(R)$ in affine coordinates as (x_R, y_R) and $(\hat{x}_R, y_R)$ respectively, and the point R in Jacobian coordinates as (X_R, Y_R, Z_R), where

$$x_R = X_R/Z_R^2, \quad y_R = Y_R/Z_R^3.$$

We denote by $\mathbf{a}$, $\mathbf{m}$, $\mathbf{m}_u$, $\mathbf{s}$, $\mathbf{s}_u$, $\mathbf{i}$, $\mathbf{r}$ and $\mathbf{z}$ the costs of addition, multiplication, multiplication without reduction, squaring, squaring without reduction, inversion, modular reduction and group exponentiation by $|z|$ in $\mathbb{F}_p$, respectively.

4.1 Faster formulas for computing small Tate pairings

We first introduce the following lemma to illustrate how to generate a basis of $E(\mathbb{F}_p)[e_2]$.

Lemma 1. *Let P be a random point of $E(\mathbb{F}_p)[e_2]$ with order e_2. Let $\omega_{e_2}(\cdot,\cdot)$ be the Weil pairing of order e_2 on E. If $\omega_{e_2}(P, \phi(P)) \neq 1$, then $E(\mathbb{F}_p)[e_2] = \langle P\rangle \oplus \langle m\phi(P)\rangle$.*

Proof. Since ϕ is an automorphism on E and $e_2 = e_1 \cdot m$, the order of $m\phi(P)$ is equal to e_1. By [43, Theorem 3.9], the condition $\omega_{e_2}(P, \phi(P)) \neq 1$ means that $\phi(P) \notin \langle P\rangle$ and thus P and $m\phi(P)$ are linearly independent. By the fact that $E(\mathbb{F}_p)[e_2] \cong \mathbb{Z}_{e_1} \oplus \mathbb{Z}_{e_2}$, we conclude that $E(\mathbb{F}_p)[e_2] = \langle P\rangle \oplus \langle m\phi(P)\rangle$. □

Based on this observation, we can compute the two Tate pairings efficiently, which is summarized as the following theorem.

Theorem 2. *Let the point P be defined as Lemma 1. Given a random non-identity point $R \in E(\mathbb{F}_p)$, then $R \in \mathbb{G}_1$ if and only if*

$$T_{e_2}(P, \hat{\phi}(R))^m = 1 \text{ and } T_{e_2}(P, R) = 1.$$

Proof. By Lemma 1, $\{P, m\phi(P)\}$ is a basis of $E(\mathbb{F}_p)[e_2]$. Then, it follows from Theorem 1 that $R \in \mathbb{G}_1$ if and only if

$$T_{e_1}(m\phi(P), R) = 1 \text{ and } T_{e_2}(P, R) = 1.$$

Sine $e_1 \mid e_2$, from [21, Exercise 26.3.8] and by the bilinearity of Tate pairing, we have

$$T_{e_1}(m\phi(P), R) = T_{e_2}(m\phi(P), R) = T_{e_2}(\phi(P), R)^m.$$

Furthermore, since $T_{e_2}(\phi(P), R) = T_{e_2}(P, \hat{\phi}(R))$, we conclude that

$$T_{e_1}(m\phi(P), R) = T_{e_2}(P, \hat{\phi}(R))^m,$$

which completes the proof of the theorem. □

Theorem 2 presents new formulas for subgroup membership testing for $\mathbb{G}_1$ in the BLS, BW13 and BW19 families. Even though it still requires computing two Tate pairings, the evaluations of the two Miller functions can be shared, significantly reducing the total number of Miller's iterations. In addition, this method benefits from low storage requirements since the Tate two pairings share the same first pairing argument.

We now introduce an alternative method tailored to the BW13 and BW19 families. First, it can be seen from Table 1 that the parameters $e_1 = (z^2-z+1)/3$ and $e_2 = (z^2 - z + 1)$ in the two families. We denote by λ_1 and λ_2 the two eigenvalues of ϕ on $E[e_1]$. It is easy to prove that

$$\lambda_1 = -z, \lambda_2 = z - 1.$$

Based on this observation, we can determine the λ_1- and λ_2- eigenspaces of the endomorphism ϕ acting on $E[e_1]$, which is summarized as the lemma below.

Lemma 2. *Let Q be a random point of $E[e_1]$ with order e_1 in the BW13 or BW19 family. Define $H_1 = E[e_1] \cap \mathrm{Ker}(\phi - \lambda_1)$ and $H_2 = E[e_1] \cap \mathrm{Ker}(\phi - \lambda_2)$. Let $Q_1 = \phi(Q) - \lambda_2 Q$ and $Q_2 = \phi(Q) - \lambda_1 Q$. Then $Q_1 \in H_1$ and $Q_2 \in H_2$. In particular, if $Q = R_1 + R_2$, where R_1 and R_2 are generators of H_1 and H_2 respectively, then $\langle Q_1 \rangle = H_1$ and $\langle Q_2 \rangle = H_2$*

Proof. By the definition of Q_1, we have

$$\phi(Q_1) = \phi^2(Q) - \lambda_2\phi(Q), \lambda_1 Q_1 = \lambda_1\phi(Q) - \lambda_1\lambda_2\phi(Q). \tag{3}$$

Since the two characteristic values λ_1 and λ_2 satisfy that

$$\lambda_1 + \lambda_2 = -1 \bmod e_1, \lambda_1\lambda_2 = 1 \bmod e_1,$$

it follows from Eq. (3) that

$$\begin{aligned}\phi(Q_1) - \lambda_1 Q_1 =& \phi^2(Q) - (\lambda_1 + \lambda_2)\phi(Q) + \lambda_1\lambda_2\phi(Q).\\ =& \phi^2(Q) + \phi(Q) + Q\\ =& \mathcal{O},\end{aligned}$$

which implies that $Q_1 \in H_1$. Moreover, since $Q = R_1 + R_2$ we get

$$Q_1 = \phi(R_1) - \lambda_2 R_1 = (\lambda_1 - \lambda_2)R_1.$$

It is straightforward to prove that

$$\frac{(-2z+1)}{3}(\lambda_1 - \lambda_2) + 4e_1 = 1.$$

In order to ensure the parameter p is an integer, the seed z must meet the condition $z \equiv 2 \bmod 3$ in the BW13 and BW19 families. Then, we have $(-2z + 1)/3 \in \mathbb{Z}$ and thus $\gcd(\lambda_1 - \lambda_2, e_1) = 1$, which implies that the order of Q_1 is equal to e_1. Thus we have $\langle Q_1 \rangle = H_1$. Likewise, we can also prove that $\langle Q_2 \rangle = H_2$, which completes the proof of the lemma. □

Let Q_3 be a point of $E(\mathbb{F}_p)$ with order $m = 3$ in the BW13 or BW19 family, and let $P_1 = Q_1, P_2 = Q_2 + Q_3$. It follows from Lemma 2 that $\{P_1, P_2\}$ is a basis of $E(\mathbb{F}_p)[e_2]$. Moreover, the points P_1 and P_2 also satisfy that

$$\phi(P_1) = -zP_1, \phi(P_2) = (z-1)Q_2 + Q_3 = (z-1)P_2. \tag{4}$$

Based on Eq. (4), we present a new formulas for computing the two small Tate pairings required for $\mathbb{G}_1$ membership testing in the BW13 and BW19 families.

Theorem 3. *Let E be a curve in the BW13 or BW19 family. Let P_1 and P_2 be defined as above. Given a random non-identity point $R \in E(\mathbb{F}_p)$, then $R \in \mathbb{G}_1$ if and only if*

$$\left(f^{z+1}_{-z,P_1}(R) \cdot f_{-z}(\hat{\phi}(R)) \cdot (y_R - y_{P_1})\right)^{(p-1)/e_1} = 1 \text{ and}$$
$$\left(f^{z+1}_{-z,P_2}(R) \cdot f_{-z}(\phi(R)) \cdot (y_R - y_{P_2})\right)^{(p-1)/e_2} = 1.$$

Proof. Since $\phi(P_1) = -zP_1$, we can deduce that

$$\begin{aligned} f_{z^2-z+1,P_1}(R) &= f_{z^2,P_1}(R) \cdot f_{-z,P_1}(R) \cdot \ell_{z^2P_1,-zP_1}(R) \\ &= f^{-z+1}_{-z,P_1}(R) \cdot f_{-z,\phi(P_1)}(R) \cdot (y_R - y_{P_1}). \end{aligned}$$

Furthermore, it follows from [44, Theorem 1] that $f_{-z,\phi(P_1)}(R) = f_{-z,P_1}(\hat{\phi}(R))$, which implies that

$$f_{z^2-z+1,P_1}(R) = f^{-z+1}_{-z,P_1}(R) \cdot f_{-z,P_1}(\hat{\phi}(R)) \cdot (y_R - y_{P_1}).$$

Thus, the e_1-order pairing $T_{e_1}(P_1, Q)$ can be expressed as

$$T_{e_1}(P_1, Q) = T_{e_2}(P_1, Q) = \left(f^{z+1}_{-z,P_1}(R) \cdot f_{-z}(\hat{\phi}(R)) \cdot (y_R - y_{P_1})\right)^{(p-1)/e_1}.$$

Similarly, we can also prove that

$$T_{e_2}(P_2, Q) = \left(f^{z+1}_{-z,P_2}(R) \cdot f_{-z}(\phi(R)) \cdot (y_R - y_{P_2})\right)^{(p-1)/e_2}.$$

Finally, the result immediately follows from Theorem 1. □

Comparison: Theorems 2 and 3 describe two distinct methods for $\mathbb{G}_1$ membership testing, both of which are suitable for the BW13 and BW19 families. Each method has its own strengths. To be precise, the first method (proposed in Theorem 2) requires less storage requirements, while the second one (proposed in Theorem 3) benefits from shorter Miller loop for a single Tate pairing computation and thus more suitable for parallel computation.

Remark 1. It should be noted that the technique proposed in Theorem 3 is not suitable for the BLS family since the action of endomorphism ϕ can not be extracted from the Miller function.

In the following, we only analyze the computational cost of the two Tate pairing computations based on the first method as it is suitable for all target pairing-friendly curves.

4.2 Miller's iteration without precomputation

Since the technique of denominator elimination [7] can not be applied to curves with embedding degree one, vertical line evaluations in Algorithm 1 cannot be ignored. To minimize these evaluations, the authors in [15] suggest performing Miller's iteration via the modified Miller function $g_{m,P}$ with divisor

$$\mathrm{div}(g_{m,P}) = m(P) + (-mP) - (m+1)(\mathcal{O}).$$

They outlined the optimal strategy for computing pairings on these curves as follows:

1. Combine two consecutive doubling steps into one quadrupling step, saving two vertical line evaluations.
2. Combine one doubling and one addition/subtraction step into a single doubling-addition/subtraction step, also saving two vertical line evaluations.

Moreover, in order to delay inversion operation in $\mathbb{F}_p$, it is necessary to update the numerators and denominators of the modified Miller function at each Miller's iteration. Based on the above observation, it was discussed in [14] that the computations the Miller function evaluations at the two points R and $\hat{\phi}(R)$ at a shared Miller loop. Specifically, the two functions $g_{m,P}(R)$ and $g_{m,P}(\hat{\phi}(R))$ can be written as

$$g_{m,P}(R) = \frac{N_m(R)}{D_m(R)}, g_{m,P}(\hat{\phi}(R)) = \frac{N_m(\hat{\phi}(R))}{D_m(\hat{\phi}(R))}.$$

Since $g_{1,P} = x - x_P$, we initially set that

$$\begin{aligned} &d_1 = N_1(R) = x_R - x_P, d_2 = N_2(R) = \hat{x}_R - x_P, d_3 = y_R - y_P, \\ &D_1(\hat{\phi}(R)) = 1 \text{ and } D_2(\hat{\phi}(R)) = 1. \end{aligned} \tag{5}$$

Observing that $f_{e_2,P} = g_{e_2-1,P}$, we actually need to compute the following four values at the shared Miller loop:

$$N_{e_2-1}(R),\ D_{e_2-1}(R),\ N_{e_2-1}((\hat{\phi}(R)) \text{ and } D_{e_2-1}((\hat{\phi}(R)).$$

This computation mainly involves the following six subroutines: `SDBL`, `SADD`, `SSUB`, `SDADD`, `SDSUB` and `SQPL`. To be precise, on the input of the tuple $(N_m(R)$, $D_m(R), N_m(\hat{\phi}(R)),\ D_m(\hat{\phi}(R), T)$ where $T = mP$, the outputs of the six subroutines are given as follows:

$$\begin{aligned}
&\texttt{SDBL}: &&(N_{2m}(R), D_{2m}(R), N_{2m}(\hat{\phi}(R)), D_{2m}(\hat{\phi}(R), 2T);\\
&\texttt{SADD}: &&(N_{m+1}(R), D_{m+1}(R), N_{m+1}(\hat{\phi}(R)), D_{m+1}(\hat{\phi}(R), T+P);\\
&\texttt{SSUB}: &&(N_{m-1}(R), D_{m-1}(R), N_{m-1}(\hat{\phi}(R)), D_{m-1}(\hat{\phi}(R), T-P);\\
&\texttt{SDADD}: &&(N_{2m+1}(R), D_{2m+1}(R), N_{2m+1}(\hat{\phi}(R)), D_{2m+1}(\hat{\phi}(R), 2T+P);\\
&\texttt{SDSUB}: &&(N_{2m-1}(R), D_{2m-1}(R), N_{2m-1}(\hat{\phi}(R)), D_{2m-1}(\hat{\phi}(R), 2T-P);\\
&\texttt{SQPL}: &&(N_{4m}(R), D_{4m}(R), N_{4m}(\hat{\phi}(R)), D_{4m}(\hat{\phi}(R), 4T).
\end{aligned}$$

Table 3: The updating functions and the precomputed values for different subroutines. The symbols λ_R and λ_{R_1,R_2} represent the slopes of the lines $\ell_{R,R}$ and ℓ_{R_1,R_2}, respectively.

subroutines	the updating functions	the precomputed values
`SDBL`	$g_{2m,P} = g_{m,P}^2 \cdot \frac{x-x_{2T}}{y+\lambda_T(x-x_{2T})-y_{2T}}$	$\lambda_T,\ x_{2T},\ y_{2T}$
`SADD`	$g_{m+1,P} = g_{m,P} \cdot \frac{y-\lambda_{T,P}(x-x_P)-y_P}{x-x_T}$	$\lambda_{T,P},\ x_T$
`SSUB`	$g_{m-1,P} = g_{m,P} \cdot \frac{x-x_{T-P}}{y-\lambda_{P,-T}(x-x_P)-y_P}$	$\lambda_{P,-T},\ x_{T-P}$
`SDADD`	$g_{2m+1,P} = g_{m,P}^2 \cdot \frac{y-y_{2T}-\lambda_{P,2T}(x-x_{2T})}{y-y_{2T}+\lambda_T(x-x_{2T})}$	$\lambda_{P,2T},\ \lambda_T,\ x_{2T},\ y_{2T}$
`SDSUB`	$g_{2m-1,P} = g_{m,P}^2 \cdot \frac{y-\lambda_{2T,-P}(x-x_{2T})-y_{2T}}{(y+\lambda_T(x-x_{2T})-y_{2T})(x-x_P)}$	$\lambda_{-P,2T},\ \lambda_T,\ x_{2T},\ y_{2T}$
`SDSUB`$_L$	$g_{2m-1,P} = \frac{g_{m,P}^2}{x-x_T}\ (2m = e_2)$	x_T
`SQPL`	$g_{4m,P} = g_{m,P}^4 \cdot \frac{(y-y_{2T}-\lambda_{2T}(x-x_{2T}))^2}{y-y_{2T}+\lambda_T(x-x_{2T})}$	$\lambda_T,\ \lambda_{2T},\ x_{2T},\ y_{2T}$

In Table 3, we present the updating functions that can be used to execute the six subroutines. The authors in [14] have demonstrated how to perform the subroutines `SADD`, `SDADD` and `SQPL`. As a supplement, we present explicit formulas of the subroutines `SDBL`, `SSUB` and `SDSUB`, which is summarized in Appendix A. Write $e_2 - 1$ in a non-adjacent form (NAF) as $e_2 - 1 = \sum_{i=0}^{l} n_i 2^i$. If $n_0 = -1$, we have the following relations:

$$g_{e_2-1,P} = \frac{g^2_{\frac{e_2}{2},P} \cdot Z^2_{\frac{e_2}{2}Q}}{x \cdot Z^2_{\frac{e_2}{2}Q} - X_{\frac{e_2}{2}Q}}.$$

Hence, it is quite convenient to perform the last iteration of the shared Miller loop as follows:

$$A = Z^2_{\frac{e_2}{2}Q}, B = A \cdot x_R - X_{\frac{e_2}{2}Q}, C = A \cdot \hat{x}_R - X_{\frac{e_2}{2}Q}, N_{e-1}(R) = N^2_{e/2}(R) \cdot A,$$
$$N_{e-1}(\hat{\phi}(R)) = N^2_{e/2}(\hat{\phi}(R)) \cdot A, D_{e-1}(R) = D^2_{e/2}(R) \cdot B, D_{e-1}(\hat{\phi}(R)) = D^2_{e/2}(\hat{\phi}(R)) \cdot C,$$

which requires $6\mathbf{m} + 5\mathbf{s} + 2\mathbf{a}$. We denote the above subroutine as `SDSUB`$_L$ such that it can be distinguished from the general `SDSUB`. In Algorithm 2, we present pseudo-code for computing $f_{e_2,P}(R)$ and $f_{e_2,P}(\hat{\phi}(R))$.

4.3 Miller's iteration with precomputation

It is well known that the computation of Miller function can be further sped up in the scenario that the first pairing argument is fixed as a system parameter. This technique was investigated by Costello and Stebila [11]. It was also applied

Algorithm 2 Shard Miller loop for two small Tate pairings

```
Input: the points $P, R \in E(\mathbb{F}_p)$, $e_2 - 1 = \sum_{i=0}^{l} n_i 2^i$ with $n_i \in \{-1, 0, 1\}$
Output: $N_1, D_1, N_2, D_2$ such that $f_{e_2,P}(R) = N_1/D_1$, $f_{e_2,P}(\hat{\phi}(R)) = N_2/D_2$
1: $N_1 \leftarrow x_R - x_P$, $D_1 \leftarrow 1$, $N_2 \leftarrow \hat{x}_R - x_P$, $D_2 \leftarrow 1$, $d_1 \leftarrow N_1$, $d_2 \leftarrow N_2$,
   $d_3 \leftarrow y_P - y_R$, $T \leftarrow P$, $i \leftarrow l - 1$
2: if $n_0 = -1$ then $j \leftarrow 1$ else $j \leftarrow 0$ end if
3: while $i \geq j$ do
4:     if $n_i = 0$ and $i \neq j$ then
5:         $T, N_1, D_1, N_2, D_2 \leftarrow$ SQPL$(T, R, N_1, D_1, N_2, D_2)$, $i \leftarrow i - 1$
6:         if $n_i = 1$ then
7:             $T, N_1, D_1, N_2, D_2 \leftarrow$ SADD$(T, R, N_1, D_1, N_2, D_2)$
8:         elif $n_i = -1$ then
9:             $T, N_1, D_1, N_2, D_2 \leftarrow$ SSUB$(T, R, N_1, D_1, N_2, D_2)$
10:        end if
11:        $i \leftarrow i - 1$
12:    elif $n_i = 1$ then
13:        $T, N_1, D_1, N_2, D_2 \leftarrow$ SDADD$(T, R, N_1, D_1, N_2, D_2)$, $i \leftarrow i - 1$
14:    elif $n_i = -1$ then
15:        $T, N_1, D_1, N_2, D_2 \leftarrow$ SDSUB$(T, R, N_1, D_1, N_2, D_2)$, $i \leftarrow i - 1$
16:    else
17:        $T, N_1, D_1, N_2, D_2 \leftarrow$ SDBL$(T, R, N_1, D_1, N_2, D_2)$, $i \leftarrow i - 1$
18:    end if
19: end while
20: if $n_0 = -1$ then
21:     $N_1, D_1, N_2, D_2 \leftarrow$ SDSUB$_L(T, R, N_1, D_1, N_2, D_2)$
22: end if
23: return $N_1, D_1, N_2, D_2$
```

to optimize the algorithm of public-key compression for isogeny-base cryptography [38]. For computing the two Miller functions $f_{e_2,P}(R)$ and $f_{e_2,P}(\hat{\phi}(R))$, we can precompute all the parameters of line functions that only depend on the public point P at the shared Miller loop. In this situation, it is convenient to use affine coordinates such that line functions can be represented in a simple form. In Table 3, we list the percomputed values across different subroutines. On this basis, in Algorithm 3 we show how to generate a lookup table `Tab` to store all the above precomputed values that required for computing $f_{e_2,P}(R)$ and $f_{e_2,P}(\hat{\phi}(R))$. As a consequence, it only needs to evaluate line functions at the two points R and $\hat{\phi}(R)$ and then accumulate them at each shared Miller's iteration. In Algorithm 4, we present pseudo-code for computing $f_{e_2,P}(R)$ and $f_{e_2,P}(\hat{\phi}(R))$ on input of the precomputed table `Tab`, the public parameter P and the candidate point R.

Algorithm 3 Generating a lookup table for pairing computation with precomputation

```
Input: the point $P \in E(\mathbb{F}_p)$, $e_2 - 1 = \sum_{i=0}^{l} n_i 2^i$ with $n_i \in \{-1, 0, 1\}$
Output: the lookup table Tab
 1: $T \leftarrow P$, $k \leftarrow 0$, $i \leftarrow l-1$,
 2: if $n_0 = -1$ then $j \leftarrow 1$ else $j \leftarrow 0$ end if
 3: while $i \geq j$ do
 4:     if $n_i = 0$ and $i \neq j$ then
 5:         Tab[$k$] $\leftarrow \lambda_T$, Tab[$k+1$] $\leftarrow \lambda_{2T}$, Tab[$k+2$] $\leftarrow x_{2T}$, Tab[$k+3$] $\leftarrow y_{2T}$
 6:         $T \leftarrow 4T$, $k \leftarrow k+4$, $i \leftarrow i-1$
 7:         if $n_i = 1$ then
 8:             Tab[$k$] $\leftarrow \lambda_{T,P}$, Tab[$k+1$] $\leftarrow x_T$, $T \leftarrow T+P$, $i \leftarrow i+1$, $k \leftarrow k+2$
 9:         elif $n_i = -1$ then
10:             Tab[$k$] $\leftarrow \lambda_{P,-T}$, Tab[$k+1$] $\leftarrow x_{T-P}$, $T \leftarrow T-P$, $i \leftarrow i+1$, $k \leftarrow k+2$
11:         end if
12:         $i \leftarrow i-1$
13:     elif $n_i = 1$ then
14:         Tab[$k$] $\leftarrow \lambda_T$, Tab[$k+1$] $\leftarrow \lambda_{P,2T}$, Tab[$k+2$] $\leftarrow x_{2T}$, Tab[$k+3$] $\leftarrow y_{2T}$
15:         $T \leftarrow 2T+P$, $k \leftarrow k+4$, $i \leftarrow i-1$
16:     elif $n_i = -1$ then
17:         Tab[$k$] $\leftarrow \lambda_T$, Tab[$k+1$] $\leftarrow \lambda_{-P,2T}$, Tab[$k+2$] $\leftarrow x_{2T}$, Tab[$k+3$] $\leftarrow y_{2T}$
18:         $T \leftarrow 2T-P$, $k \leftarrow k+4$, $i \leftarrow i-1$
19:     else
20:         Tab[$k$] $\leftarrow \lambda_T$, Tab[$k+1$] $\leftarrow x_{2T}$, Tab[$k+2$] $\leftarrow y_{2T}$, $T \leftarrow 2T$, $k \leftarrow k+3$, $i \leftarrow i-1$
21:     end if
22: end while
23: if $n_0 = -1$ then Tab [$k$] $\leftarrow x_T$ end if
24: return Tab
```

4.4 The final exponentiation

In Algorithm 5, we summarize the process of subgroup membership testing for $\mathbb{G}_1$ on our target curves. It should be noted that line evaluations vanish at the candidate point R or $\hat{\phi}(R)$ only if $R \in \langle P \rangle$. Hence, one or more of the four updated values in Algorithms 2 and 4 might be equal to zero. In this case, the testing can be aborted early. Otherwise, we continue to performing the final exponentiation part. In this phase, we first use the trick of Montgomery simultaneous inversion [37] to compute $f_1 = f_{e_2,P}(\hat{\phi}(R))$ and $f_2 = f_{e_2,P}(R)$ such that one inversion operation in $\mathbb{F}_p$ can be saved,

$$f_1 = \frac{N_1 \cdot D_2}{D_1 \cdot D_2}, f_2 = \frac{N_2 \cdot D_1}{D_1 \cdot D_2}. \tag{6}$$

Then the computation of the two Tate pairings can be done by raising f_1 and f_2 to the power of $\exp_1 = (p-1)/e_1$ and $\exp_2 = (p-1)/e_2$, respectively. We

Algorithm 4 Shard Miller loop for two small Tate pairings with precomputation

```
Input: the lookup table Tab, the points $P, R \in E(\mathbb{F}_p)$, $e_2 - 1 = \sum_{i=0}^{l} n_i 2^i$ with $n_i \in \{-1, 0, 1\}$
Output: $N_1, D_1, N_2, D_2$ such that $f_{e_2,P}(R) = N_1/D_1$, $f_{e_2,P}(\hat{\phi}(R)) = N_2/D_2$
1: $N_1 \leftarrow x_R - x_P$, $D_1 \leftarrow 1$, $N_2 \leftarrow \hat{x}_R - x_P$, $D_2 \leftarrow 1$, $d_1 \leftarrow N_1$, $d_2 \leftarrow N_2$, $d_3 \leftarrow y_R - y_P$, $i \leftarrow l - 1$
2: if $n_0 = -1$ then $j \leftarrow 1$ else $j \leftarrow 0$ end if
3: while $i \geq j$ do
4:     if $n_i = 0$ and $i \neq j$ then
5:         $t_0 \leftarrow x_R - \texttt{Tab}[k+2]$, $t_1 \leftarrow \hat{x}_R - \texttt{Tab}[k+2]$, $t_2 \leftarrow y_R - \texttt{Tab}[k+3]$
6:         $t_3 \leftarrow t_2 + t_0 \cdot \texttt{Tab}[k]$, $t_4 \leftarrow t_2 + t_1 \cdot \texttt{Tab}[k]$, $t_5 \leftarrow t_2 - t_0 \cdot \texttt{Tab}[k+1]$
7:         $t_6 \leftarrow t_2 + t_1 \cdot \texttt{Tab}[k+1]$, $D_1 \leftarrow D_1^4 \cdot t_3$, $D_2 \leftarrow D_2^4 \cdot t_4$, $N_1 \leftarrow (N_1^2 \cdot t_5)^2$
8:         $N_2 \leftarrow (N_2^2 \cdot t_6)^2$, $i \leftarrow i - 1$, $k \leftarrow k + 4$
9:         if $n_i = 1$ then
10:            $t_0 \leftarrow x_R - \texttt{Tab}[k+1]$, $t_1 \leftarrow \hat{x}_R - \texttt{Tab}[k+1]$, $t_2 \leftarrow d_1 \cdot \texttt{Tab}[k]$
11:            $t_3 \leftarrow d_2 \cdot \texttt{Tab}[k]$, $D_1 \leftarrow D_1 \cdot t_0$, $D_2 \leftarrow D_2 \cdot t_1$, $N_1 \leftarrow N_1 \cdot (d_3 - t_2)$
12:            $N_2 \leftarrow N_1 \cdot (d_3 - t_3)$, $i \leftarrow i - 1$, $k \leftarrow k + 2$
13:        elif $n_i = -1$ then
14:            $t_0 \leftarrow x_R - \texttt{Tab}[k+1]$, $t_1 \leftarrow \hat{x}_R - \texttt{Tab}[k+1]$, $t_2 \leftarrow d_1 \cdot \texttt{Tab}[k]$
15:            $t_3 \leftarrow d_2 \cdot \texttt{Tab}[k]$, $N_1 \leftarrow N_1 \cdot t_0$, $N_2 \leftarrow N_2 \cdot t_1$, $D_1 \leftarrow D_1 \cdot (d_3 - t_2)$
16:            $D_2 \leftarrow D_2 \cdot (d_3 - t_3)$, $i \leftarrow i - 1$, $k \leftarrow k + 2$
17:        end if
18:        $i \leftarrow i - 1$
19:    elif $n_i = 1$ then
20:        $t_0 \leftarrow x_R - \texttt{Tab}[k+2]$, $t_1 \leftarrow \hat{x}_R - \texttt{Tab}[k+2]$, $t_2 \leftarrow y_R - \texttt{Tab}[k+3]$
21:        $t_3 \leftarrow t_2 + t_0 \cdot \texttt{Tab}[k]$, $t_4 \leftarrow t_2 + t_1 \cdot \texttt{Tab}[k]$, $t_5 \leftarrow t_2 - t_0 \cdot \texttt{Tab}[k+1]$
22:        $t_6 \leftarrow t_2 - t_1 \cdot \texttt{Tab}[k+1]$, $D_1 \leftarrow D_1^2 \cdot t_3$, $D_2 \leftarrow D_2^2 \cdot t_4$, $N_1 \leftarrow N_1^2 \cdot t_5$
23:        $N_2 \leftarrow N_2^2 \cdot t_6$, $i \leftarrow i - 1$, $k \leftarrow k + 4$
24:    elif $n_i = -1$ then
25:        $t_0 \leftarrow x_R - \texttt{Tab}[k+2]$, $t_1 \leftarrow \hat{x}_R - \texttt{Tab}[k+2]$, $t_2 \leftarrow y_R - \texttt{Tab}[k+3]$
26:        $t_3 \leftarrow (t_2 + t_0 \cdot \texttt{Tab}[k]) \cdot d_1$, $D_1 \leftarrow D_1^2 \cdot t_3$, $t_4 \leftarrow (t_2 + t_1 \cdot \texttt{Tab}[k]) \cdot d_2$
27:        $D_2 \leftarrow D_2^2 \cdot t_4$, $t_5 \leftarrow t_2 - t_0 \cdot \texttt{Tab}[k+1]$, $t_6 \leftarrow t_2 - t_1 \cdot \texttt{Tab}[k+1]$
28:        $N_1 \leftarrow N_1^2 \cdot t_5$, $N_2 \leftarrow N_2^2 \cdot t_6$, $i \leftarrow i - 1$, $k \leftarrow k + 4$
29:    else
30:        $t_0 \leftarrow x_R - \texttt{Tab}[k+1]$, $t_1 \leftarrow \hat{x}_R - \texttt{Tab}[k+1]$, $t_2 \leftarrow y_R - \texttt{Tab}[k+2]$
31:        $t_3 \leftarrow t_2 + t_0 \cdot \texttt{Tab}[k]$, $t_4 \leftarrow t_2 + t_1 \cdot \texttt{Tab}[k]$, $N_1 \leftarrow N_1^2 \cdot t_0$, $N_2 \leftarrow N_2^2 \cdot t_1$
32:        $D_1 \leftarrow D_1^2 \cdot t_3$, $D_2 \leftarrow D_2^2 \cdot t_4$, $i \leftarrow i - 1$, $k \leftarrow k + 3$
33:    end if
34: end while
35: if $n_0 = -1$ then
36:     $t_0 \leftarrow x_R - \texttt{Tab}[k]$, $t_1 \leftarrow \hat{x}_R - \texttt{Tab}[k]$, $N_1 \leftarrow N_1^2$, $N_2 \leftarrow N_2^2$, $D_1 \leftarrow D_1^2 \cdot t_0$
37:     $D_2 \leftarrow D_2^2 \cdot t_1$
38: end if
39: return $N_1, D_1, N_2, D_2$
```

Algorithm 5 subgroup membership testing for $\mathbb{G}_1$ in the BLS, BW13 and BW19 families

```
Input: the candidate point $R \in E(\mathbb{F}_p)$
Output: a bit 0 or 1. If $R \in \mathbb{G}_1$, then it returns 1; otherwise, it returns 0.
 1: Computing $N_1$, $D_1$, $N_2$, $D_2$ by Algorithm 2 or Algorithm 4
 2: if $N_1 = 0$ or $D_1 = 0$ or $N_2 = 0$ or $D_2 = 0$ then
 3:     return 0
 4: else
 5:     $h \leftarrow (D_1 \cdot D_2)^{-1}$
 6:     $f_1 \leftarrow (h \cdot D_2 \cdot N_1)^{(p-1)/e_1}$
 7:     $f_2 \leftarrow (h \cdot D_1 \cdot N_2)^{(p-1)/e_2}$
 8:     if $f_1 = f_2 = 1$ then
 9:         return 1
10:     else
11:         return 0
12:     end if
13: end if
```

notice that the first exponent $\exp_1$ can be parameterized by a polynomial with small integral coefficients for our target curves. For example, the exponent $\exp_1$ in the BLS12 family is given as

$$\exp_1 = |z^5 - z^4 - z^3 + z^2 + z + 2|.$$

Since it is sufficient to determine whether the pairing values are equal to 1 or not for $\mathbb{G}_1$ membership testing, the computation of the first final exponentiation can be replaced by checking that

$$\begin{cases} f_1^{z^5+z^2+z+2} = f_1^{z^4+z^3}, & \text{if } z > 0, \\ f_1^{-z^5+z^4-z} = f_1^{-z^3+z^2+2}, & \text{if } z < 0. \end{cases}$$

In Table 4, we present the exponent $\exp_1$ and the cost of the final exponentiation by $\exp_1$ in the BLS, BW13 and BW19 families. However, when representing $\exp_2$ in the basis of z, we find that the coefficients are not small. For example, the exponent $\exp_2$ in the BLS12 family is given as

$$\exp_2 = |81z_0^5 + 108z_0^4 + 45z_0^3 + 6z_0^2 + z_0 + 1|,$$

where $z_0 = (z-1)/3$. Thus, it seems more efficient to compute $f_2^{\exp_2}$ directly. One might attempt to accelerate the computation of the second final exponentiation by raising the second pairing to the power of m such that the new exponent can be also represented as a polynomial with small integral coefficients. Unfortunately, it could result in invalid points passing the membership testing. For instance, given $R' = R_1' + R_2'$ where $R_1' \in E(\mathbb{F}_p)[m]$ and $R_2' \in \mathbb{G}_1$, it is straight-

Table 4: The exponent $\exp_1$ in the BLS, BW13 and BW19 families. The symbol $\mathbf{e}_1$ represents the cost of the exponentiation by $\exp_1$.

family	$\exp_1$	$\mathbf{e}_1$
BW13	$\lvert z^{26} - z^{13} - 3z^{12} - 3z^{11} + 3z^9 + 3z^8 - 3z^6 - 3z^5 + 3z^3 + 3z^2 - 2\rvert$	$26\mathbf{z} + 11\mathbf{m} + 3\mathbf{s}$
BW19	$\lvert z^{38} - z^{19} - 3z^{18} - 3z^{17} + 3z^{15} + 3z^{14} - 3z^{12} - 3z^{11} + 3z^9 + 3z^8 - 3z^6 - 3z^5 + 3z^3 + 3z^2 - 2\rvert$	$38\mathbf{z} + 15\mathbf{m} + 3\mathbf{s}$
BLS12	$\lvert z^5 - z^4 - z^3 + z^2 + z + 2\rvert$	$5\mathbf{z} + 4\mathbf{m} + \mathbf{s}$
BLS24	$\lvert z^9 - z^8 - z^5 + z^4 + z + 2\rvert$	$9\mathbf{z} + 4\mathbf{m} + \mathbf{s}$
BLS48	$\lvert z^{17} - z^{16} - z^9 + z^8 + z + 2\rvert$	$17\mathbf{z} + 4\mathbf{m} + \mathbf{s}$

forward to see that

$$T_{e_2}(P, R')^m = T_{e_2}(P, R'_2)^m = T_{e_2}(P, mR'_2) = 1,$$
$$T_{e_2}(P, \hat{\phi}(R'))^m = T_{e_2}(P, \hat{\phi}(R'_2))^m = T_{e_2}(P, m\hat{\phi}(R'_2)) = 1.$$

4.5 Computational cost

Let n_1, n_2, n_3, n_4, n_5, n_6, and n_7 denote the number of the subroutines `SDBL`, `SADD`, `SSUB`, `SDADD`, `SDSUB`, $\texttt{SDSUB}_L$ and `SQPL` in the execution of the shared Miller loop. In Table 5, we summarize the operation counts of these subroutines on ordinary curves with j-invariant 0. Assuming that the final exopnentiation by $\exp_2$ is performed using the windowed squaring-and-multiplication algorithm, then the total cost of $\mathbb{G}_1$ membership testing of the proposed algorithm is approximately

$$\underbrace{n_1 \cdot \texttt{SDBL} + n_2 \cdot \texttt{SADD} + n_3 \cdot \texttt{SSUB} + n_4 \cdot \texttt{SDADD} + n_5 \cdot \texttt{SDSUB} + n_6 \cdot \texttt{SDSUB}_L + n_7 \cdot \texttt{SQPL}}_{\text{main part of the shared Miller loop}}$$
$$+ \underbrace{\mathbf{i} + 5\mathbf{m} + 3\mathbf{a}}_{\text{Eqs. 5 and 6}} + \underbrace{\mathbf{e}_1 + (\texttt{nbits}(\exp_2) - 1)\mathbf{s} + \frac{\texttt{nbits}(\exp_2)}{(w+1)}\mathbf{m} + (2^{w-1} - 1)\mathbf{m} + \mathbf{s}}_{\text{two final exponentiations}},$$

where $\texttt{nbits}(\exp_2)$ represents the bit length of $\exp_2$ and w is the selected window size. By selecting $w = 4$, we estimate the costs of $\mathbb{G}_1$ membership testings on the seven candidate pairing-friendly curves, which is summarized in Table 6.

Table 5: Costs of different subroutines required for computing $f_{e_2,P}(R)$ and $f_{e_2,P}(\hat{\phi}(R))$ for the seven candidate pairing-friendly curves.

subroutine	without precomputation	with precomputation
`SDBL`	$11\mathbf{m} + 4\mathbf{m}_u + 8\mathbf{s} + \mathbf{s}_u + 15\mathbf{a} + 3\mathbf{r}$	$6\mathbf{m} + 4\mathbf{s} + 5\mathbf{a}$
`SADD`	$15\mathbf{m} + 5\mathbf{m}_u + 3\mathbf{s} + 15\mathbf{a} + 3\mathbf{r}$	$6\mathbf{m} + 4\mathbf{a}$
`SSUB`	$14\mathbf{m} + 5\mathbf{m}_u + 4\mathbf{s} + 14\mathbf{a} + 3\mathbf{r}$	$6\mathbf{m} + 4\mathbf{a}$
`SDADD`	$16\mathbf{m} + 8\mathbf{m}_u + 10\mathbf{s} + \mathbf{s}_u + 26\mathbf{a} + 6\mathbf{r}$	$8\mathbf{m} + 4\mathbf{s} + 7\mathbf{a}$
`SDSUB`	$18\mathbf{m} + 8\mathbf{m}_u + 10\mathbf{s} + \mathbf{s}_u + 26\mathbf{a} + 6\mathbf{r}$	$10\mathbf{m} + 4\mathbf{s} + 7\mathbf{a}$
$\texttt{SDSUB}_L$	$6\mathbf{m} + 5\mathbf{s} + 2\mathbf{a}$	$2\mathbf{m} + 4\mathbf{s} + 2\mathbf{a}$
`SQPL`	$14\mathbf{m} + 7\mathbf{m}_u + 15\mathbf{s} + 2\mathbf{s}_u + 28\mathbf{a} + 6\mathbf{r}$	$8\mathbf{m} + 8\mathbf{s} + 7\mathbf{a}$

Table 6: Costs of subgroup membership testings for $\mathbb{G}_1$ on the seven candidate pairing-friendly curves.

curve	$(n_1, n_2, \cdots, n_7)$	$\mathbf{z}$	without precomputation	with precomputation
BLS12-381	$(0,2,0,3,0,0,30)$	$5\mathbf{m} + 63\mathbf{s}$	$597\mathbf{m} + 244\mathbf{m}_u + 1119\mathbf{s}$ $+63\mathbf{s}_u + 948\mathbf{a} + 204\mathbf{r}$	$375\mathbf{m} + 885\mathbf{s} + 239\mathbf{a}$
BLS12-446	$(0,2,0,4,0,0,35)$	$6\mathbf{m} + 74\mathbf{s}$	$699\mathbf{m} + 287\mathbf{m}_u + 1314\mathbf{s}$ $+74\mathbf{s}_u + 1114\mathbf{a} + 240\mathbf{r}$	$439\mathbf{m} + 1039\mathbf{s} + 281\mathbf{a}$
BW13-310	$(0,2,2,0,2,0,10)$	$3\mathbf{m} + 11\mathbf{s}$	$387\mathbf{m} + 106\mathbf{m}_u + 761\mathbf{s}$ $+22\mathbf{s}_u + 390\mathbf{a} + 84\mathbf{r}$	$277\mathbf{m} + 665\mathbf{s} + 100\mathbf{a}$
BW19-286	$(1,2,3,1,0,0,6)$	$2\mathbf{m} + 7\mathbf{s}$	$335\mathbf{m} + 79\mathbf{m}_u + 667\mathbf{s}$ $+14\mathbf{s}_u + 281\mathbf{a} + 60\mathbf{r}$	$244\mathbf{m} + 597\mathbf{s} + 74\mathbf{a}$
BLS24-315	$(1,1,1,1,0,1,14)$	$10\mathbf{m} + 31\mathbf{s}$	$415\mathbf{m} + 120\mathbf{m}_u + 803\mathbf{s}$ $+30\mathbf{s}_u + 464\mathbf{a} + 99\mathbf{r}$	$297\mathbf{m} + 687\mathbf{s} + 120\mathbf{a}$
BLS24-509	$(0,0,0,2,1,0,24)$	$9\mathbf{m} + 50\mathbf{s}$	$569\mathbf{m} + 192\mathbf{m}_u + 1299\mathbf{s}$ $+51\mathbf{s}_u + 750\mathbf{a} + 162\mathbf{r}$	$401\mathbf{m} + 1113\mathbf{s} + 189\mathbf{a}$
BLS48-575	$(1,0,3,0,1,0,15)$	$9\mathbf{m} + 31\mathbf{s}$	$553\mathbf{m} + 132\mathbf{m}_u + 1326\mathbf{s}$ $+32\mathbf{s}_u + 503\mathbf{a} + 108\mathbf{r}$	$426\mathbf{m} + 1199\mathbf{s} + 129\mathbf{a}$

5 Implementation Results

We first present Magma code to verify the correctness of our proposed algorithms and formulas. In order to compare the performance between our proposed algorithms and the previous state-of-the-art techniques, we also provide high-speed software implementation within the RELIC toolkit, which is a well known cryptographic library that mainly implemented in the C programming language with

ASM acceleration for the lower prime field arithmetic. The library provides the state-of-the-art implementations of pairing group operations on different pairing-friendly curves, including all the curves listed in Table 2 except BW13-310 and BW19-286. Recently, Dai et al. [14] also used RELIC to implement pairing group operations on BW13-310. Thus, we have integrated our code into RELIC to ensure fair performance comparisons. All of the benchmarks were taken on an 3.00GHZ Intel(R) Core(TM) i7-9700 CPU running at Ubuntu 22.04 LTS averaged over 10^4 executions with the TurboBoost disabled and HyperThreading turned off. The main compiler used was GCC version 11.4.0, with optimization `flags-O3-funroll-loops-march=native -mtune=native`. In **Fig**. 1, we present

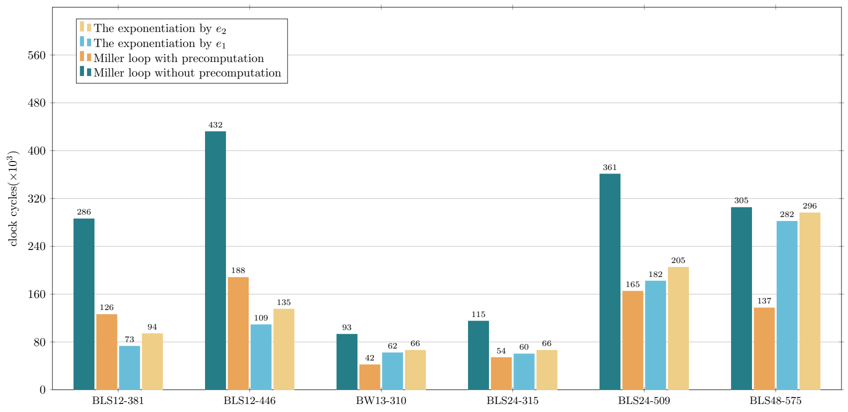

Fig. 1: Timings of each building blocks for two small Tate pairing computations on a list of pairing-friendly curves.

the timing result (measured in $\times 10^3$ clock cycles) of each building block for two small Tate pairing computations on our target curves. In **Fig**. 2, we compare the performance of $\mathbb{G}_1$ membership testing on these curves between our work and the previous leading work. The results show that our method without prcomputation is about 62.0%, 22.4%, 46.2% and 80.6% faster than the previous fastest method on the BW13-310, BLS24-315, BLS24-509 and BLS48-575 curves, respectively. However, it is around 14.6% and 41.4% slower on the BLS12-381 and BLS12-446 curves. Moreover, the performance advantage of our method can be further extended with prcomputation. In this case, our method is about 34.8%, 10.6%, 110.6%, 63.9%, 98.1% and 123.1% faster than the previous fastest method on the BLS12-381, BLS12-446, BW13-310, BLS24-315, BLS24-509 and BLS48-575 curves, respectively. It should be noted that the previous fastest method can not be sped up via percomputation.

To summarize, our method is well-suited for curves with a small value of ρ ($\rho = \log p / \log r \geq 1$), such as BW13-310 and BLS48-575. Indeed, by the fact that $e_1 \approx e_2$ and $p \approx e_1 \cdot e_2 \cdot r$ for our chosen curves, it is easy to deduce that $2\log e_2/\log r \approx \rho - 1$. Recall that the previous fastest method requires approximately $\log r/2$ group operations, while our one involves around $\log e_2$ Miller's iterations and two group exponentiations in $\mathbb{F}_p$. Thus, the value of $\rho - 1$ approximately represents the ratio of the computational costs between the two methods.

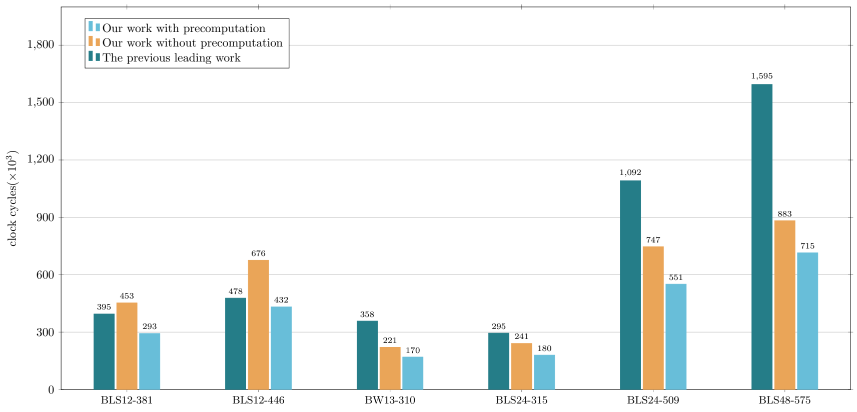

Fig. 2: Timings of $\mathbb{G}_1$ membership testing on a list of pairing-friendly curves between our work and the previous leading work.

6 Conclusion

In this paper, we revisited subgroup membership testing for $\mathbb{G}_1$ on pairing-friendly curves via the Tate pairing. We first introduced faster formulas that suitable for the BLS, BW13 and BW19 families such that the computations of the two Tate pairings only require around $\log e_2$ Miller's iterations. The new formulas also benefit from less storage requirements. Moreover, we also provided a high performance software implementation for our proposed algorithm and compared it to the previous leading work across several popular pairing-friendly curves. Our results exhibited a significant performance advantage over the previous fastest one on BW13-310, BLS24-315, BLS24-509 and BLS48-575. With precomputation, our method also outperforms the previous fastest one on BLS12-381 and BLS12-446.

References

1. Aranha, D.F., Gouvêa, C.P.L.: Relic is an efficient library for cryptography, https://github.com/relic-toolkit/relic
2. Aranha, D.F., Fotiadis, G., Guillevic, A.: A short-list of pairing-friendly curves resistant to the special TNFS algorithm at the 192-bit security level. Cryptology ePrint Archive, Paper 2024/1223 (2024), https://eprint.iacr.org/2024/1223

3. Aranha, D.F., Hvass, B.S., Spitters, B., Tibouchi, M.: Faster constant-time evaluation of the kronecker symbol with application to elliptic curve hashing. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security – CCS 2023. p. 3228–3238. Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3576915.3616597
4. Azarderakhsh, R., Fishbein, D., Grewal, G., Hu, S., Jao, D., Longa, P., Verma, R.: Fast software implementations of bilinear pairings. IEEE Transactions on Dependable and Secure Computing **14**(6), 605–619 (2017). https://doi.org/10.1109/TDSC.2015.2507120
5. Barbulescu, R., Gaudry, P., Kleinjung, T.: The tower number field sieve. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015. pp. 31–55. Springer Berlin Heidelberg, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_2
6. Barreto, P.S.L.M., Costello, C., Misoczki, R., Naehrig, M., Pereira, G.C.C.F., Zanon, G.: Subgroup security in pairing-based cryptography. In: Lauter, K., Rodríguez-Henríquez, F. (eds.) Progress in Cryptology – LATINCRYPT 2015. pp. 245–265. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-22174-8_14
7. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: Yung, M. (ed.) Advances in Cryptology — CRYPTO 2002. pp. 354–369. Springer Berlin Heidelberg, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_23
8. Barreto, P.S.L.M., Lynn, B., Scott, M.: On the selection of pairing-friendly groups. In: Matsui, M., Zuccherato, R.J. (eds.) Selected Areas in Cryptography – SAC 2003. pp. 17–25. Springer Berlin Heidelberg, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24654-1_2
9. Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Designs, Codes and Cryptography **37**(1), 133–141 (2005). https://doi.org/10.1007/s10623-004-3808-4
10. Clarisse, R., Duquesne, S., Sanders, O.: Curves with fast computations in the first pairing group. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) Cryptology and Network Security. pp. 280–298. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-65411-5_14
11. Costello, C., Stebila, D.: Fixed argument pairings. In: Abdalla, M., Barreto, P.S.L.M. (eds.) Progress in Cryptology – LATINCRYPT 2010. pp. 92–108. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14712-8_6
12. Dai, Y., He, D., Peng, C., Yang, Z., an Zhao, C.: Revisiting pairing-friendly curves with embedding degrees 10 and 14. Cryptology ePrint Archive, Paper 2023/1958 (2023), `https://eprint.iacr.org/2023/1958`
13. Dai, Y., Lin, K., Zhao, C.A., Zhou, Z.: Fast subgroup membership testings for $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ on pairing-friendly curves. Designs, Codes and Cryptography **91**(10), 3141–3166 (2023). https://doi.org/10.1007/s10623-023-01223-7
14. Dai, Y., Zhang, F., Zhao, C.A.: Don't forget pairing-friendly curves with odd prime embedding degrees. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(4), 393–419 (2023). https://doi.org/10.46586/tches.v2023.i4.393-419
15. Dai, Y., Zhou, Z., Zhang, F., Zhao, C.A.: Software implementation of optimal pairings on elliptic curves with odd prime embedding degrees. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **105**(5), 858–870 (2022). https://doi.org/10.1587/transfun.2021EAP1115

16. El Housni, Y., Guillevic, A.: Families of SNARK-friendly 2-chains of elliptic curves. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022. pp. 367–396. Springer International Publishing, Cham (2022)
17. El Housni, Y., Guillevic, A., Piellard, T.: Co-factor clearing and subgroup membership testing on pairing-friendly curves. In: Batina, L., Daemen, J. (eds.) Progress in Cryptology – AFRICACRYPT 2022. pp. 518–536. Springer Nature Switzerland, Cham (2022). https://doi.org/10.1007/978-3-031-17433-9_22
18. Electric Coin Company: What is jubjub?, https://bitzecbzc.github.io/technology/jubjub
19. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Journal of Cryptology **23**(2), 224–280 (2010)
20. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953 (2019), https://eprint.iacr.org/2019/953
21. Galbraith, S.: Mathematics of Public Key Cryptography. Cambridge University Press (2018), https://www.math.auckland.ac.nz/~sgal018/crypto-book/main.pdf, version 2
22. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 190–200. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
23. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016. pp. 305–326. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_11
24. Guillevic, A.: A short-list of pairing-friendly curves resistant to special TNFS at the 128-bit security level. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography – PKC 2020. pp. 535–564. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_19
25. Guillevic, A.: Pairing-friendly curves (2021), https://members.loria.fr/AGuillevic/pairing-friendly-curves/
26. Hamburg, M.: Ed448-goldilocks, a new elliptic curve. Cryptology ePrint Archive, Paper 2015/625 (2015), https://eprint.iacr.org/2015/625
27. Hamburg, M.: Computing the jacobi symbol using Bernstein-Yang. Cryptology ePrint Archive, Paper 2021/1271 (2021), https://eprint.iacr.org/2021/1271
28. Joux, A., Pierrot, C.: The special number field sieve in $\mathbb{F}_{p^n}$. In: Cao, Z., Zhang, F. (eds.) Pairing-Based Cryptography – Pairing 2013. pp. 45–61. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-04873-4_3
29. Joye, M., Lapiha, O., Nguyen, K., Naccache, D.: The eleventh power residue symbol. Journal of Mathematical Cryptology **15**(1), 111–122 (2020). https://doi.org/10.1515/jmc-2020-0077
30. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010. pp. 177–194. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_11
31. Kim, T., Barbulescu, R.: Extended tower number field sieve: A new complexity for the medium prime case. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016. pp. 543–571. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_20

32. Koshelev, D.: Subgroup membership testing on elliptic curves via the Tate pairing. Journal of Cryptographic Engineering **13**(1), 125–128 (Apr 2023). https://doi.org/10.1007/s13389-022-00296-9
33. Koshelev, D.: Correction to: Subgroup membership testing on elliptic curves via the Tate pairing. Journal of Cryptographic Engineering **14**(1), 127–128 (Apr 2024). https://doi.org/10.1007/s13389-023-00331-3
34. Masson, S., Sanso, A., Zhang, Z.: Bandersnatch: a fast elliptic curve built over the BLS12-381 scalar field. Designs, Codes and Cryptography (2024). https://doi.org/10.1007/s10623-024-01472-0
35. Mbiang, N.B., Aranha, D.D.F., Fouotsa, E.: Computing the optimal ate pairing over elliptic curves with embedding degrees 54 and 48 at the 256-bit security level. International Journal of Applied Cryptography **4**(1), 45–59 (2020). https://doi.org/10.1504/IJACT.2020.107167
36. Miller, V.S.: The Weil pairing, and its efficient calculation. Journal of Cryptology **17**(4), 235–261 (2004). https://doi.org/10.1007/s00145-004-0315-8
37. Montgomery, P.L.: Speeding the pollard and elliptic curve methods of factorization. Mathematics of computation **48**(177), 243–264 (1987)
38. Naehrig, M., Renes, J.: Dual isogenies and their application to public-key compression for isogeny-based cryptography. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 243–272. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-34621-8_9
39. Schoof, R.: Nonsingular plane cubic curves over finite fields. Journal of combinatorial theory, Series A **46**(2), 183–211 (1987). https://doi.org/10.1016/0097-3165(87)90003-3
40. Scott, M.: Miracl–multiprecision integer and rational arithmetic c/c++ library.
41. Scott, M.: A note on group membership tests for $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ on BLS pairing-friendly curves. Cryptology ePrint Archive, Report 2021/1130 (2021), `https://ia.cr/2021/1130`
42. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory **56**(1), 455–461 (2009). https://doi.org/10.1109/TIT.2009.2034881
43. Washington, L.: Elliptic curves: Number theory and cryptography, second edition. CRC Press, Boca Raton (2008)
44. Zhao, C.A., Zhang, F., Huang, J.: A note on the ate pairing. International Journal of Information Security **7**(6), 379–382 (2008)

A Explicit Formulas

Notations. We use the notation $\times$ to denote field multiplication without reduction. We write for λ_R and λ_{R_1,R_2} the slopes of the lines $\ell_{R,R}$ and ℓ_{R_1,R_2}, respectively.

In this section, we analyze the computational costs of the subroutines `SDBL`, `SSUB` and `SDSUB` on ordinary curves with j-invariant 0. Let $T = mP$ be in Jacobian coordinates for some point P and non-zero integer m. Using the formulas provided in [4, Setion 4.3], the point $2T$ can be computed via the sequence of operations

$$A = X_T^2, B = A/2, C = A + B, D = C^2, E = Y_T^2, F = X_T \cdot E, X_{2T} = D - 2F,$$
$$U_0 = C \times (F - X_{2T}), U_1 = E \times E, Y_{2T} = (U_0 - U_1) \bmod p, Z_{2T} = Y_T \cdot Z_T.$$

Assuming that the computation of $U_0 - U_1$ requires $2\mathbf{a}$, then the total cost of the point doubling is $2\mathbf{m} + \mathbf{m}_u + 3\mathbf{s} + \mathbf{s}_u + 7\mathbf{a} + \mathbf{r}$. If $P \neq T$, the authors of [4] also provide explicit formulas to compute $T + P$ by a mixed addition:

$$A = Z_T^2, \theta = y_P \cdot A \cdot Z_T - Y_T, \beta = x_P \cdot A - X_T, B = \beta^2, C = \beta \cdot B, D = X_T \cdot B, Z_{T+P} = Z_T \cdot \beta,$$
$$X_{T+P} = \theta^2 - 2D - C, U_0 = \theta \times (D - X_{T+P}), U_1 = Y_T \times C, Y_{T+P} = (U_0 - U_1) \bmod p,$$

which comes at a cost of $6\mathbf{m} + 2\mathbf{m}_u + 3\mathbf{s} + 8\mathbf{a} + \mathbf{r}$.

A.1 `SDBL`

The modified Miller function $g_{2m,P}$ can be obtained from $g_{m,P}$ as follows:

$$g_{2m,P} = g_{m,P}^2 \cdot \frac{x - x_{2T}}{y - \lambda_{-T}(x - x_{2T}) - y_{2T}}.$$

Writing the point $2T$ in Jacobian coordinates, then the functions $N_{2m}(x, y)$ and $D_{2m}(x, y)$ can be expressed as

$$N_{2m}(x, y) = N_m^2(x, y) \cdot Z_{2T} \cdot (x \cdot Z_{2T}^2 - X_{2T}),$$
$$D_{2m}(x, y) = D_m^2(x, y) \cdot (yZ_{2T}^3 - Y_{2T} + 3/2X_T^2 \cdot (x \cdot Z_{2T}^2 - X_{2T})).$$

We first compute the point $2T = (X_{2T}, Y_{2T}, Z_{2T})$. Then the values $N_{2m}(R)$, $D_{2m}(R)$, $N_{2m}(\hat{\phi}(R))$ and $D_{2m}(\hat{\phi}(R))$ can be obtained by performing the following sequence of operations at a costs of $9\mathbf{m} + 3\mathbf{m}_u + 5\mathbf{s} + 8\mathbf{a} + 2\mathbf{r}$:

$$A = Z_{2T}^2, B = A \cdot Z_{2T}, C_1 = A \cdot x_R - X_{2T}, C_2 = A \cdot \hat{x}_R - X_{2T}, L_1 = C_1 \cdot Z_{2T},$$
$$L_3 = C_2 \cdot Z_{2T}, U_0 = y_R \times B, U_1 = \frac{3}{2}X_T^2 \times C_1, U_2 = \frac{3}{2}X_T^2 \times C_2, E = (U_0 + U_1) \bmod p,$$
$$F = (U_0 + U_2) \bmod p, L_2 = E - Y_{2T}, L_4 = F - Y_{2T}, N_{2m}(R) = N_m^2(R) \cdot L_1,$$
$$D_{2m}(R) = D_m^2(R) \cdot L_2, N_{2m}(\hat{\phi}(R)) = N_m^2(\hat{\phi}(R)) \cdot L_3, D_{2m}(\hat{\phi}(R)) = D_m^2(\hat{\phi}(R)) \cdot L_4,$$

where $\frac{3}{2}X_T^2$ is given during the computation of $2T$. In total, the subroutine `SDBL` requires $11\mathbf{m} + 4\mathbf{m}_u + 8\mathbf{s} + \mathbf{s}_u + 15\mathbf{a} + 3\mathbf{r}$.

A.2 `SSUB`

The modified Miller function $g_{m-1,P}$ can be obtained from $g_{m,P}$ as follows:

$$g_{m-1,P} = g_{m,P} \cdot \frac{x - x_{T-P}}{y - \lambda_{-T,P}(x - x_P) - y_P}.$$

Then, the two functions $N_{m-1}(x, y)$ and $D_{m-1}(x, y)$ can be expressed as

$$N_{m-1}(x, y) = N_m(x, y) \cdot (x \cdot Z_{T-P}^2 - X_{T-P}),$$
$$D_{m-1}(x, y) = D_m(x, y) \cdot Z_{T-P} \cdot \big((y - y_P) \cdot Z_{T-P} + \theta_{T-P} \cdot (x - x_P)\big),$$

where $\theta_{T-P} = -y_P \cdot Z_T^3 - Y_T$ can be obtained during the computation of $T - P$. Thus, the values of $N_{m-1}(R)$, $D_{m-1}(R)$, $N_{m-1}(\hat{\phi}(R))$ and $D_{m-1}(\hat{\phi}(R))$ can be computed via the following sequence of operations at a cost of $8\mathbf{m} + 3\mathbf{m}_u + \mathbf{s} + 6\mathbf{a} + 2\mathbf{r}$:

$$A = Z_{T-P}^2, L_1 = x_R \cdot A - X_{T-P}, L_3 = \hat{x}_R \cdot A - X_{T-P}, U_0 = d_3 \times Z_{T-P}, U_1 = d_1 \times \theta_{T-P},$$
$$U_2 = d_2 \times \theta_{T-P}, B = (U_0 + U_1) \bmod p, C = (U_0 + U_2) \bmod p, L_2 = B \cdot Z_{T-P},$$
$$L_4 = C \cdot Z_{T-P}, N_{m-1}(\hat{\phi}(R)) = N_m(\hat{\phi}(R)) \cdot L_3, D_{m-1}(\hat{\phi}(R)) = D_m(\hat{\phi}(R)) \cdot L_4,$$
$$N_{m-1}(R) = N_m(R) \cdot L_1, D_{m-1}(R) = D_m(R) \cdot L_2,$$

where d_1, d_2 and d_3 are given at the initial stage of the Miller loop (Line 1 in Algorithm 2). In total, the subroutine `SSUB` requires $14\mathbf{m} + 5\mathbf{m}_u + 4\mathbf{s} + 14\mathbf{a} + 3\mathbf{r}$.

A.3 `SDSUB`

The modified Miller functions $g_{m,P}$ and $g_{2m-1,P}$ satisfy the following relation:

$$g_{2m-1,P} = g_{m,P}^2 \cdot \frac{y - \lambda_{2T,-P}(x - x_{2T}) - y_{2T}}{(y - \lambda_{-T,-T}(x - x_{2T}) - y_{2T})(x - x_P)}.$$

Then, the two functions $N_{2m-1}(x, y)$ and $D_{2m-1}(x, y)$ can be expressed as

$$N_{2m-1}(x, y) = N_m^2(x, y)\big((yZ_{2T}^3 - Y_{2T}) \cdot \beta_{2T-P} - (xZ_{2T}^2 - X_{2T}) \cdot \theta_{2T-P}\big),$$
$$D_{2m-1}(x, y) = D_m^2(x, y) \cdot (x - x_P) \cdot \beta_{2T-P} \cdot \Big((yZ_{2T}^3 - Y_{2T}) + \frac{3}{2}X_T^2(xZ_{2T}^2 - X_{2T})\Big),$$

where $\beta_{2T-P} = x_P \cdot Z_{2T}^2 - X_{2T}, \theta_{2T-P} = -y_P \cdot Z_{2T}^3 - Y_{2T}$. In order to compute $N_{2m-1}(R)$, $D_{2m-1}(R)$, $N_{2m-1}(\hat{\phi}(R))$ and $D_{2m-1}(\hat{\phi}(R))$, we first compute $2T = (X_{2T}, Y_{2T}, Z_{2T})$ and $2T - P = (X_{2T-P}, Y_{2T-P}, Z_{2T-P})$. During this process, the intermediate variables $Z_{2T}^2, Z_{2T}^3, \beta_{2T-P}, \theta_{2T-P}$ and $\frac{3}{2}X_T^2$ can be obtained. Thus, we compute the above four updated values by performing the following sequence of operations:

$$A_1 = x_R \cdot Z_{2T}^2 - X_{2T}, A_2 = \hat{x}_R \cdot Z_{2T}^2 - X_{2T}, B = y_R \cdot Z_{2T}^3 - Y_{2T}, C = \frac{3}{2}X_T^2 \cdot \beta_{2T-P},$$
$$U_0 = B \times \beta_{2T-P}, U_1 = A_1 \times \theta_{2T-P}, U_2 = A_2 \times \theta_{2T-P}, U_3 = C \times A_1, U_4 = C \times A_2,$$
$$E = (U_0 + U_3) \bmod p, F = (U_0 + U_4) \bmod p, L_1 = (U_0 - U_1) \bmod p, L_3 = (U_0 - U_2) \bmod p,$$
$$L_2 = d_1 \cdot E, L_4 = d_2 \cdot F, N_{2m-1}(R) = N_m^2(R) \cdot L_1, D_{2m-1}(R) = D_m^2(R) \cdot L_2,$$
$$N_{2m-1}(\hat{\phi}(R)) = N_m^2(\hat{\phi}(R)) \cdot L_3, D_{2m-1}(\hat{\phi}(R)) = D_m^2(\hat{\phi}(R)) \cdot L_4,$$

which requires $10\mathbf{m} + 5\mathbf{m}_u + 4\mathbf{s} + 11\mathbf{a} + 4\mathbf{r}$. In total, the subroutine `SDSUB` can be executed at a cost of $18\mathbf{m} + 8\mathbf{m}_u + 10\mathbf{s} + \mathbf{s}_u + 26\mathbf{a} + 6\mathbf{r}$.