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

Yu Dai[1], Debiao He[2]✉, Dimitri 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 introduced an efficient method of subgroup membership testing for a list of non-pairing-friendly curves, using at most two small Tate pairings. In fact, this technique can also be applied to certain pairing-friendly curves, e.g., from the BLS and BW13 families. In this paper, we revisit Koshelev's method and propose simplified formulas for computing the two Tate pairings. Compared to the original formulas, ours reduce both the number of Miller's iterations and the storage requirements. Furthermore, we provide a high-speed software implementation on a 64-bit processor. Our experimental results show that the new method outperforms the state-of-the-art one by up to 62.0% and 41.2% on the BW13-310 and BLS48-575 curves, respectively. When special precomputation is utilized, our method achieves greater speed improvements of up to 110.6% and 74.4% on the two curves, respectively.

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

1 Introduction

A cryptographic pairing on an elliptic curve E over a large prime field $\mathbb{F}_p$ is a bilinear and non-degenerate map of the form $\mathbb{G}_1 \oplus \mathbb{G}_2 \to \mathbb{G}_T$, 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 a subgroup of the 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 [24] and PlonK [22]. In pairing-based cryptographic protocols, participants often need to exponentiate in one or more pairing groups. However, these groups typically lie in larger groups 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 exponentiation on an element of non-prime order with a secret key, this may expose partial information about the secret key. We refer the reader to [6] for more details.

In order to resist small subgroup attacks in pairing-based cryptographic protocols, it is essential to verify 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 [42] proposed an efficient method of subgroup membership testing on the BLS12 curves via an easily computable endomorphism. His method is approximately $2\times$, $4\times$ 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. [14] generalized Scott's technique such that it can be applied to a large class of pairing-friendly curves. In more detail, Dai et al.'s method requires about $\log_2(r)/2$, $\log_2(r)/\varphi(k)$ and $\log_2(r)/\varphi(k)$ 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 $\log_2(r)/(2\varphi(k))$ on some special pairing-friendly curves [13]. It should also be noted that the cost of $\mathbb{G}_2$ membership testing may be almost free if it is allowed to be executed during pairing computation.

Recently, Koshelev [33] invented a novel method of subgroup membership testing for non-pairing-friendly (i.e., plain) curves via the *Tate pairing*. His method imposes a specific restriction on the group structure of 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 [44, Theorem 4.1] that $E(\mathbb{F}_p) \cong \mathbb{Z}_{e_1} \oplus \mathbb{Z}_{e_2 \cdot r}$ for uniquely defined naturals 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 the two Tate pairings of orders e_1 and e_2. In particular, if $E(\mathbb{F}_p)$ is cyclic, i.e., $e_1 = 1$, it only requires evaluating one e_2-order pairing. This technique is well-suited for zk-SNARK-friendly curves like Jubjub [20] and Bandersnatch [35]. Since $e_1 \in \{1, 2\}$ and $e_2 \in \{2, 8\}$ for them, the lengths of the Miller loops are extremely short, meaning that the whole pairing computations almost amount to their final exponentiations. Interestingly, the latter can be further accelerated when $e_2 \leq 11$ according to [30]. It should be noted that Koshelev [34] subsequently generalized his method such that it can be applied to certain curves with $e_2 \nmid (p-1)$, e.g., Ed448-Goldilocks [27]. 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 technique is also suitable for $\mathbb{G}_1$ membership testing on a list of mainstream pairing-friendly curves, such as the Barreto-Lynn-Scott(BLS) family [8] and the complete families from [21, Construction 6.6] with embedding degrees 13 and 19. The latter are also referred to as BW curves in the literature, as they are constructed by means of the Brezing-Weng method [10]. However, when applying the idea of Koshelev to these curves (unlike the aforementioned plain ones), the computational cost cannot be ignored anymore. As an example, the sizes of e_1 and e_2 are about $4\times$ smaller than the size of r for the BLS12 curves. This makes Koshelev's method in its original form more expensive than Scott's method.

1.1 Our contributions

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

- We obtain an efficient algorithm for simultaneously computing two peculiar Tate pairings, i.e., a shared Miller loop supplemented by two independent final exponentiations.
- 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 algorithm achieves speed improvements of up to 62.0%, 6.5%, 2.7% and 41.2% on the BW13-310, BLS24-315, BLS24-509 and BLS48-575 curves, respectively, while being approximately 50.5% and 48.6% slower on the BLS12-381 and BLS12-446 curves, respectively;
  2. when precomputation is utilized, our algorithm is approximately 2.7%, 5.3%, 110.6%, 41.4%, 40.2% and 74.4% faster than the previous leading one 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 with a focus on those appropriate for pairings. Section 3 surveys existing methods (and a new variant of Koshelev's one) for subgroup membership testing on elliptic curves. Explicit formulas and algorithms concretizing the new variant (for the desired BLS, BW13 and BW19 curves) are contained in Section 4. 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 the Tate pairing. Then, we introduce a series of parameterized pairing-friendly families.

2.1 Elliptic curves and pairings

Let $\mathbb{F}_p$ be a large prime field. 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 $j(E) = 1728 \cdot 4a^3/(4a^3 + 27b^2)$. The group $E(\mathbb{F}_p)$ consists of $\mathbb{F}_p$-rational points (x, y) that satisfy the curve equation, together with the point at infinity $\mathcal{O} = (0 : 1 : 0)$. The order of $E(\mathbb{F}_p)$ is equal to $\#E(\mathbb{F}_p) = p + 1 - t$, 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 $r \parallel \#E(\mathbb{F}_p)$. Then, the group $E(\mathbb{F}_p) \cong \mathbb{Z}_{e_1} \oplus \mathbb{Z}_{e_2 \cdot r}$ for e_1, $e_2 \in \mathbb{N}$ such that $m = e_2/e_1 \in \mathbb{N}$. The embedding degree k of E with respect to r and p is the smallest degree such that $r \mid (p^k - 1)$. If $k > 1$, then it is equally the smallest degree such that the r-torsion group $E[r] \subseteq E(\mathbb{F}_{p^k})$. In this case, $E[r]$ is the direct product of the r-order subgroups $\mathbb{G}_1 = E(\mathbb{F}_p)[r]$ and $\mathbb{G}_2 = E(\mathbb{F}_{p^k})[r] \cap \mathrm{Ker}(\pi - [p])$. A cryptographic pairing on E is a non-degenerate bilinear map (mainly) of the form $\mathbb{G}_1 \oplus \mathbb{G}_2 \to \mathbb{G}_T$, where the codomain is the subgroup of r-th roots of unity in $\mathbb{F}_{p^k}^*$.

Given $i \in \mathbb{Z}$ and $P \in E$, we let $f_{i,P}$ be the normalized rational function on E with divisor

$$\mathrm{div}(f_{i,P}) = i(P) - (iP) - (i - 1)(\mathcal{O}).$$

Hereafter, we refer to $f_{i,P}$ as Miller functions. Let n be an integer such that $n \mid (p - 1)$. The reduced Tate pairing of order n on $E(\mathbb{F}_p)$ is defined as

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

where μ_n is the subgroup of n-th roots of unity in $\mathbb{F}_p^*$. The Tate pairing T_n 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 $P = \mathcal{O}$. Similarly, 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 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., $\alpha \circ \hat{\alpha} = [\deg(\alpha)]$.

The computation of the Tate pairing $T_n(P, R)$ can be divided into two phases: the Miller loop [37] (Algorithm 1) and the final exponentiation. In more detail, the first phase involves evaluating $f_{n,P}(R)$ and the second one aims to raise $f_{n,P}(R)$ to the power of $(p - 1)/n$. Let $\ell_{iP,jP}$ be the line through iP and jP (tangent to E at the point if $iP = jP$), and ν_{iP} be the vertical line through iP. The value $f_{n,P}(R)$ is recursively determined in the algorithm via the equation

$$f_{i+j,P} = f_{i,P} \cdot f_{j,P} \cdot \mu_{iP,jP} \tag{1}$$

Algorithm 1 Miller's algorithm.

Input: $P \in E(\mathbb{F}_p)[n]$, $R \in E(\mathbb{F}_p) \setminus \{\mathcal{O}\}$ and $n = \sum_{i=0}^{\ell} n_i 2^i$ with $n_i \in \{0,1\}$, $n_\ell = 1$

Output: $f_{n,P}(R)$

```
1: $T \leftarrow P$, $f \leftarrow 1$
2: for $i = \ell - 1$ down to $0$ do
3:     $f \leftarrow f^2 \cdot \mu_{T,T}(R)$, $T \leftarrow 2T$
4:     if $n_i = 1$ then
5:         $f \leftarrow f \cdot \mu_{T,P}(R)$, $T \leftarrow T + P$
6:     end if
7: end for
8: return $f$
```

with the auxiliary function

$$\mu_{iP,jP} = \begin{cases} \dfrac{\ell_{iP,jP}}{\nu_{(i+j)P}} & \text{if} \quad (i+j)P \neq \mathcal{O}, \\ \nu_P & \text{otherwise.} \end{cases}$$

Throughout this paper, we assume that the curve E is ordinary with j-invariant 0. In this case, the coefficient $a = 0$ and $p \equiv 1 \bmod 3$ due to [44, Proposition 4.33]. Therefore, there exists the easily computable endomorphism $\phi : (x, y) \to (\omega x, y)$ on E, where ω is a primitive cube root of unity in $\mathbb{F}_p^*$. This map is also called the GLV endomorphism as it was used by Gallant, Lambert and Vanstone [23] to accelerate elliptic curve scalar multiplication. The dual of ϕ is given as $\hat{\phi} : (x, y) \to (\omega^2 x, y)$. Their characteristic polynomial is obviously of the form $X^2 + X + 1$. Moreover, the curve E has complex multiplication by $\mathbb{Z}[\phi]$, which is a maximal quadratic order. Hence, the integer e_1 is the largest integer such that $e_1^2 \mid \#E(\mathbb{F}_p)$ and $e_1 \mid (p-1)$ (see [40, Proposition 3.7]).

2.2 Parameterized families of pairing-friendly curves

Pairing-friendly curves are specifically designed to facilitate high-performance pairing implementations at the required security levels. These curves typically have a low embedding degree k and a small value $\rho = \log_2(p)/\log_2(r) \gtrsim 1$. Table 1 summarizes some popular parameterized families of pairing-friendly curves with $1.11 \lesssim \rho \lesssim 1.5$ and embedding degree k ranging from 12 to 48, including the BLS12, BW13, BW19, BLS24 and BLS48 families. All their curves have the j-invariant 0, meaning that they can be defined by the equation $y^2 = x^3 + b$ for some $b \in \mathbb{F}_p^*$. Among other things, the quotient $m = 3$ for each family under consideration, but it is reasonable for universality to continue writing the symbol m. There may be families with another e_2-torsion structure, although still relevant to pairing-based $\mathbb{G}_1$ membership testing: the condition $m \mid (p-1)$ does not generally imply that $m = 3$.

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 [5,29,32], 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 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 the 128-bit, 192-bit and 256-bit security levels, respectively [2, 25, 36]; BW13-310 and BW19-286 provide good performance of exponentiation in $\mathbb{G}_1$ at the 128-bit security level [11]; BLS24-315 [18] is adequate for constructing zk-SNARKs based on KZG [31] polynomial commitment, e.g., PLONK [22]. For more information on selecting pairing-friendly curves, we refer to Guillevic's blog [26].

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

family-k	p	r	t	e_1	e_2	m
BLS12	$\frac{1}{3}(z-1)^2(z^4-z^2+1)+z$	$\Phi_{12}(z)$	$z+1$	$\frac{\lvert z-1\rvert}{3}$	$\lvert z-1\rvert$	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	$\frac{1}{3}(z-1)^2(z^8-z^4+1)+z$	$\Phi_{24}(z)$	$z+1$	$\frac{\lvert z-1\rvert}{3}$	$\lvert z-1\rvert$	3
BLS48	$\frac{1}{3}(z-1)^2(z^{16}-z^8+1)+z$	$\Phi_{48}(z)$	$z+1$	$\frac{\lvert z-1\rvert}{3}$	$\lvert z-1\rvert$	3

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

curve	b	z	$\lceil\log_2(p)\rceil$	$\lceil\log_2(r)\rceil$	$\rho\approx$	$\lceil\log_2(p^k)\rceil$
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

3 Efficient methods for subgroup membership testing on elliptic curves

In this section, we discuss recognized approaches in the existing literature for testing the membership to subgroups on elliptic curves.

3.1 Method I: subgroup membership testing via an easily computable endomorphism

Scott [42] invented the first non-trivial method for $\mathbb{G}_1$ membership testing specifically designed for the BLS12 family. After that, El Housni, Guillevic and Piellard [19] confirmed that this technique is also suitable for the BLS24 and BLS48 families. In 2023, Dai et al. [14] further generalized Scott's method such that it can be applied to a large class of pairing-friendly curves. In essence, they are demanded to be equipped with a cheap endomorphism. In particular, for ordinary curves with j-invariant 0, Dai et al.'s method can be summarized as follows.

Theorem 1 ([14, Theorem 3]). *Let E be an ordinary curve over $\mathbb{F}_p$ with j-invariant 0, and let the* GLV *endomorphism ϕ on $\mathbb{G}_1$ act as a scalar multiplication by λ_1. Define the two-dimensional lattice*

$$\mathcal{L}_\phi = \{(u_0, u_1) \in \mathbb{Z}^2 \mid u_0 + u_1\lambda_1 \equiv 0 \bmod r\}.$$

Consider a vector $u \in \mathcal{L}_\phi$ such that

$$\gcd(\#E(\mathbb{F}_p),\ u_0^2 - u_0u_1 + u_1^2) = r. \tag{2}$$

Finally, let R be a point in $E(\mathbb{F}_p)$. Then, $R \in \mathbb{G}_1$ if and only if

$$u_0R + u_1\phi(R) = \mathcal{O}.$$

For efficiency, we expect that the infinity norm $\|u\|_\infty$ is as small as possible. According to [43, Theorem 7], there exists a short vector $v \in \mathcal{L}_\phi$ such that $\|v\|_\infty \approx \sqrt{r}$. Fortunately, the condition (2) is generally mild, allowing the target vector u to be selected as v for many popular pairing-friendly curves. Furthermore, the two scalars u_0 and u_1 are frequently related such that we actually deal with one usual scalar multiplication. Consequently, the Scott(-type) method requires approximately $\log_2(r)/2$ 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 [33] proposed a new insight into subgroup membership testing on elliptic curves. The essence of his method is captured in the following theorem.

Theorem 2 ([33, Lemma 1]). *Let E be an elliptic curve over $\mathbb{F}_p$ with the group structure $E(\mathbb{F}_p) \cong \mathbb{Z}_{e_1} \oplus \mathbb{Z}_{e_2 \cdot r}$ and $e_2 \mid (p-1)$. Let P_1 and P_2 be two points with orders e_1 and e_2, respectively, such that $E(\mathbb{F}_p)[e_2] = \langle P_1 \rangle \oplus \langle P_2 \rangle$. Finally, let R be yet another point in $E(\mathbb{F}_p)$. Then, $R \in \mathbb{G}_1$ if and only if*

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

According to Theorem 2, subgroup membership testing on elliptic curves can be accomplished at a cost of the two Tate pairings of orders e_1 and e_2. In fact, if $E(\mathbb{F}_p)$ is cyclic, i.e., $e_1 = 1$, the method only needs computing the e_2-order Tate pairing. In addition, Koshelev also demonstrated that if $e_i \leq 11$, then the final exponentiation part of the e_i-order Tate pairing can be dramatically accelerated by means of an Euclidean-type algorithm [30]. In particular, if $e_i = 2$, then the final exponentiation is equivalent to determining the Legendre symbol, which can be further optimized using the algorithms presented in [3, 28].

In Theorem 2, it is not necessary to restrict $\{P_1, P_2\}$ to form a basis of $E(\mathbb{F}_p)[e_2]$. Instead, it suffices for the two points to generate this group.

Proposition 1. *Let P_1 and P_2 be two points that generate the group $E(\mathbb{F}_p)[e_2]$, and let R be yet another point in $E(\mathbb{F}_p)$. Then, $R \in \mathbb{G}_1$ if and only if*

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

Proof. It is nearly identical to the corresponding proof of [33, Lemma 1]. □

3.3 Summary

The above two methods differ significantly in terms of applicability, efficiency and storage requirements. Specifically, Method I is mostly tailored to pairing-friendly curves, while Method II was originally developed for plain curves such as Jubjub [20] and Bandersnatch [35]. To sum up, the main differences between the two approaches are:

1. Method I is well-suited for elliptic curves equipped with an efficiently computable endomorphism, while Method II is applicable when the curve parameters meet the condition $e_2 \mid (p-1)$.
2. The cost of Method I arises from (two-)scalar multiplication with length $\approx \log_2(r)/2$ bits. In contrast, the cost of Method II comes from two Tate pairings, the longest of which has the length $\approx \log_2(e_2)$ bits.
3. Method I does not occupy additional memory since the coefficient ω of the endomorphism ϕ is initially provided to accelerate scalar multiplication in $\mathbb{G}_1$, involved in pairing-based protocols themselves. As a comparison, Method II requires storing two points P_1 and P_2 that generate $E(\mathbb{F}_p)[e_2]$.

It is clear that Method I is suitable for the BLS, BW13 and BW19 families. Fortunately, we notice that they 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 loops is non-negligible. In addition, the technique of [30] cannot be exploited to speed up the final exponentiations as e_1, $e_2 \gg 11$. To the best of our knowledge, before the present work, Method I has thereby been the state of the art as regards subgroup membership testing (for $\mathbb{G}_1$) on pairing-friendly curves. That is why this method was implemented in numerous cryptographic libraries, including MIRACL [41] and RELIC [1].

3.4 New variant of Method II

In this section, we revisit Method II in such a way that it is still applied to the BLS, BW13 and BW19 families, becoming less costly (even than Method I in a series of cases). We first formulate the next lemma to illustrate how to obtain the generators of $E(\mathbb{F}_p)[e_2]$.

Lemma 1. *Let P be a point of $E(\mathbb{F}_p)$ with order e_2, and let $\tilde{m}$ be an integer such that $\gcd(e_1, \tilde{m}) = 1$. Let $W_{e_2}(\cdot,\cdot)$ be the Weil pairing of order e_2 on E. If additionally the order of $W_{e_2}(P, \phi(P))$ is equal to e_1, then $\{P,\, \tilde{m}\phi(P)\}$ is a pair of generators for $E(\mathbb{F}_p)[e_2]$.*

Proof. For convenience, we define the auxiliary group homomorphism

$$\begin{aligned}\Psi : \mathbb{Z}_{e_1} \oplus \mathbb{Z}_{e_2} &\to E(\mathbb{F}_p)[e_2] \\ (\ell_1, \ell_2) &\to \ell_1 \tilde{m}\phi(P) + \ell_2 P.\end{aligned}$$

The pair $\{P,\, \tilde{m}\phi(P)\}$ generates $E(\mathbb{F}_p)[e_2]$ if and only if the map Ψ is surjective. Since $\#E(\mathbb{F}_p)[e_2] = e_1e_2$, it then suffices to show that Ψ is injective, i.e., $\ker(\Psi) = \{(0,0)\}$. By definition, for any $(\ell_1, \ell_2) \in \ker(\Psi)$, we have:

$$\Psi(\ell_1, \ell_2) = \ell_1\tilde{m}\phi(P) + \ell_2 P = \mathcal{O}. \tag{3}$$

Then, it is straightforward to see that

$$W_{e_2}(P, \phi(P))^{\ell_1\tilde{m}} = W_{e_2}(P, \ell_1\tilde{m}\phi(P)) = W_{e_2}(P, \Psi(\ell_1, \ell_2)) = 1.$$

Since $W_{e_2}(P, \phi(P))$ has order e_1, $\gcd(e_1, \tilde{m}) = 1$ and $\ell_1 \in \mathbb{Z}_{e_1}$, we can deduce that $\ell_1 = 0$. Furthermore, since $\ell_2 \in \mathbb{Z}_{e_2}$ and the point P has order e_2, it implies from Equation (3) that $\ell_2 = 0$. Thus, we conclude that $\ker(\Psi) = \{(0,0)\}$, which completes the proof of the lemma. □

Given a random e_2-order point $P \in E(\mathbb{F}_p)$, our experience demonstrates that $W_{e_2}(P, \phi(P))$ has the order e_1 with high probability. At least, such a point exists for every curve from Table 2. Based on this observation, we establish a viable variant of Method II, which is summarized in the following theorem.

Theorem 3. *Let the point P and integer $\tilde{m}$ be defined as in Lemma 1, and let R be yet another point in $E(\mathbb{F}_p)$. Then, $R \in \mathbb{G}_1$ if and only if*

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

Proof. By Lemma 1, the two points P and $\tilde{m}\phi(P)$ can generate $E(\mathbb{F}_p)[e_2]$. Then, it implies from Proposition 1 that $R \in \mathbb{G}_1$ if and only if

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

By the bilinearity and compatibility of Tate pairing, we have:

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

which completes the proof of the theorem. □

Remark 1. If $\gcd(e_1, m) = 1$ with $m = e_2/e_1$, nothing prevents from choosing $\tilde{m} = m$ such that the final exponentiation part for one of the Tate pairings can be performed more efficiently (see Section 4.3 for details); otherwise, the choice $\tilde{m} = 1$ is always available.

Theorem 3 presents a new approach for testing the membership to the subgroup $\mathbb{G}_1$. Even though it still requires computing two Tate pairings, the evaluations of the corresponding Miller functions can be shared, significantly reducing the total number of Miller's iterations. In addition, this approach benefits from lower storage requirements due to the same first pairing argument.

4 Explicit formulas and algorithms

In this section, we formalize the aforementioned Koshelev-type method and analyze its computational cost.

Notations. We represent the points R and $\hat{\phi}(R)$ in affine coordinates as (x_R, y_R) and $(\hat{x}_R, y_R)$, and the point R in Jacobian coordinates as (X_R, Y_R, Z_R), where $x_R = X_R/Z_R^2$ and $y_R = Y_R/Z_R^3$. We write λ_R and λ_{R_1,R_2} for the slopes of the lines $\ell_{R,R}$ and ℓ_{R_1,R_2}. Finally, we denote by $\mathbf{i}$, $\mathbf{m}$, $\mathbf{m}_u$, $\mathbf{s}$, $\mathbf{s}_u$, $\mathbf{a}$ and $\mathbf{r}$ the costs of inversion, multiplication, multiplication without reduction $\times$, squaring, squaring without reduction, addition and reduction itself, respectively.

4.1 Miller's iterations without precomputation

Since the technique of denominator elimination [7] is not applicable to curves with embedding degree one, vertical line evaluations in Algorithm 1 cannot be ignored. In this case, the formulas proposed in [9,16] can be employed to minimize the number of vertical line evaluations. Specifically, the authors of [16] suggest performing Miller's iterations via modified Miller functions $g_{i,P}$ (given $i \in \mathbb{Z}$ and $P \in E$) with divisors

$$\text{div}(g_{i,P}) = i(P) + (-iP) - (i+1)(\mathcal{O}).$$

They outlined the optimal strategy for computing pairings on curves with odd prime embedding degrees:

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 the inversion operation in $\mathbb{F}_p$, it is necessary to update the numerator and denominator of the function $g_{i,P}$ at each Miller's iteration.

Based on the above observation, Dai et al. [15] discussed how to evaluate a Miller function at the two points R and $\hat{\phi}(R)$ in a shared Miller loop. Specifically, the two values $g_{i,P}(R)$ and $g_{i,P}(\hat{\phi}(R))$ can be written as

$$g_{i,P}(R) = \frac{N_i(R)}{D_i(R)}, \qquad g_{i,P}(\hat{\phi}(R)) = \frac{N_i(\hat{\phi}(R))}{D_i(\hat{\phi}(R))}.$$

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

$$\begin{aligned} &d_1 = N_1(R) = x_R - x_P, \quad d_2 = N_2(\hat{\phi}(R)) = \hat{x}_R - x_P, \quad d_3 = y_R - y_P, \\ &D_1(R) = 1, \quad D_2(\hat{\phi}(R)) = 1. \end{aligned} \tag{4}$$

For conciseness, we put

$$\tau_i(R) = \big(N_i(R),\ D_i(R),\ N_i(\hat{\phi}(R)),\ D_i(\hat{\phi}(R))\big).$$

Observing that $f_{e_2,P} = g_{e_2-1,P}$, we actually need to determine the tuple $\tau_{e_2-1}(R)$. This computation mainly involves six subroutines: `SDBL`, `SADD`, `SSUB`, `SDADD`, `SDSUB` and `SQPL`. To be precise, on input $\tau_i(R)$ and T, where $T = iP$, the outputs of these subroutines are given in Table 3.

Table 3: The outputs of the subroutines.

`SDBL`	`SADD`	`SSUB`	`SDADD`	`SDSUB`	`SQPL`
$\tau_{2i}(R)$, $2T$	$\tau_{i+1}(R)$, $T+P$	$\tau_{i-1}(R)$, $T-P$	$\tau_{2i+1}(R)$, $2T+P$	$\tau_{2i-1}(R)$, $2T-P$	$\tau_{4i}(R)$, $4T$

Table 4 is dedicated to updating functions that are used to execute the defined subroutines. The authors of [15] demonstrated in detail how to perform `SADD`, `SDADD` and `SQPL`. As a supplement, we summarize in Appendix A explicit operation sequences for the remaining subroutines `SDBL`, `SSUB` and `SDSUB`. In Algorithm 2, we present pseudo-code for computing $f_{e_2,P}(R)$ and $f_{e_2,P}(\hat{\phi}(R))$, whose input consists of the fixed point P from Lemma 1, the candidate point R and the number $e_2 - 1 = \sum_{i=0}^{\ell} n_i 2^i$ in its non-adjacent form (NAF).

Table 4: The updating functions and the precomputed values for the defined subroutines.

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

Remark 2. If $n_0 = -1$, we have the relation

$$g_{e_2-1,P} = \frac{g_{\tilde{e}_2,P}^2 \cdot Z_{\tilde{e}_2 Q}^2}{x \cdot Z_{\tilde{e}_2 Q}^2 - X_{\tilde{e}_2 Q}},$$

where $\tilde{e}_2 = e_2/2$ is an integer. Hence, it is quite convenient to finish the last iteration of the shared Miller loop as follows:

$$A = Z_{\tilde{e}_2 Q}^2, \quad B = A \cdot x_R - X_{\tilde{e}_2 Q}, \quad C = A \cdot \hat{x}_R - X_{\tilde{e}_2 Q}, \quad N_{e_2-1}(R) = N_{\tilde{e}_2}^2(R) \cdot A,$$
$$N_{e_2-1}(\hat{\phi}(R)) = N_{\tilde{e}_2}^2(\hat{\phi}(R)) \cdot A, \quad D_{e_2-1}(R) = D_{\tilde{e}_2}^2(R) \cdot B, \quad D_{e_2-1}(\hat{\phi}(R)) = D_{\tilde{e}_2}^2(\hat{\phi}(R)) \cdot C,$$

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`.

4.2 Miller's iterations with precomputation

It is well known that the evaluation of a Miller function can be further sped up in the scenario when the first pairing argument is fixed as a system parameter. This technique was investigated by Costello and Stebila [12]. It was also applied to optimize the algorithm of public-key compression for isogeny-based cryptography [39]. Consequently, for computing the Miller function $f_{e_2,P}$ at the points R and $\phi(R)$, we can predetermine all the coefficients of line functions that only

Algorithm 2 The shared Miller loop for the two Tate pairings without precomputation.

```
Input: $P \in E(\mathbb{F}_p)[e_2]$, $R \in E(\mathbb{F}_p) \setminus \{\mathcal{O}\}$ and $e_2 - 1 = \sum_{i=0}^{\ell} n_i 2^i$ with $n_i \in \{-1, 0, 1\}$, $n_\ell = 1$
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: $N_1 \leftarrow x_R - x_P$, $D_1 \leftarrow 1$, $N_2 \leftarrow \hat{x}_R - x_P$, $D_2 \leftarrow 1$, $d_1 \leftarrow N_1$, $d_2 \leftarrow N_2$,
   $d_3 \leftarrow y_P - y_R$, $T \leftarrow P$, $i \leftarrow \ell - 1$
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 $N_1, D_1, N_2, D_2$
```

depend on the point P. In this situation, it is convenient to use affine coordinates such that the line functions can be represented in a simple form.

In Table 4, we list precomputed values across the introduced subroutines. In Algorithm 3, we show how to generate a lookup table `Tab` to store all these values required for finding $f_{e_2,P}(R)$ and $f_{e_2,P}(\hat{\phi}(R))$. Thus, in the shared Miller loop, one only needs to evaluate the line functions at R and $\hat{\phi}(R)$, accumulating properly their results. In Algorithm 4, we present the corresponding pseudo-code whose input includes the table `Tab`.

Remark 3. In the case of precomputation, we observe that it is more efficient to use the technique proposed in [17, Section 6.2] to execute the subroutines `SDADD` and `SDSUB`. Therefore, the modified Miller functions $g_{2i+1,P}$ and $g_{2i-1,P}$ can be

Algorithm 3 Generating the lookup table for pairing evaluation with precomputation.

```
Input: $P \in E(\mathbb{F}_p)[e_2]$ and $e_2 - 1 = \sum_{i=0}^{\ell} n_i 2^i$ with $n_i \in \{-1, 0, 1\}$, $n_\ell = 1$
Output: Tab
 1: $T \leftarrow P$, $k \leftarrow 0$, $i \leftarrow \ell - 1$
 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[$k$]$\leftarrow x_T$, Tab[$k+1$]$\leftarrow y_T$,Tab[$k+2$]$\leftarrow x_T + x_{T+P} + \lambda_{T,P} \cdot \lambda_{T+P,T}$
15:         Tab[$k+3$]$\leftarrow \lambda_{T,P} + \lambda_{T+P,T}$, $T \leftarrow 2T + P$, $k \leftarrow k+4$, $i \leftarrow i-1$
16:     elif $n_i = -1$ then
17:         Tab[$k$]$\leftarrow x_T$, Tab[$k+1$]$\leftarrow y_T$,Tab[$k+2$]$\leftarrow x_T + x_{T-P} + \lambda_{T,-P} \cdot \lambda_{T-P,T}$
18:         Tab[$k+3$]$\leftarrow \lambda_{T,-P} + \lambda_{T-P,T}$, $T \leftarrow 2T - P$, $k \leftarrow k+4$, $i \leftarrow i-1$
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
```

written as follows:

$$g_{2i+1,P} = \frac{g_{i,P}^2}{\nu_T^2} \cdot \frac{\ell_{T,P} \cdot \ell_{T+P,T}}{\nu_{T+P}}, \qquad g_{2i-1,P} = \frac{g_{i,P}^2}{\nu_T^2 \cdot \nu_P} \cdot \frac{\ell_{T,-P} \cdot \ell_{T-P,T}}{\nu_{T-P}},$$

where $\ell_{T,\pm P} \cdot \ell_{T\pm P,T}/\nu_{T\pm P}$ can be expressed as the parabola

$$(x - x_T)(x + x_T + x_{T\pm P} + \lambda_{T,\pm P} \cdot \lambda_{T\pm P,T}) - (\lambda_{T,\pm P} + \lambda_{T\pm P,T})(y - y_T).$$

4.3 Final exponentiations

In Algorithm 5, we summarize the process of subgroup membership testing for $\mathbb{G}_1$, grounded on the new variant of Method II. 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 may be equal to zero. In this case, the testing can be aborted early.

Algorithm 4 The shared Miller loop for the two Tate pairings with precomputation.

Input: Tab, $P \in E(\mathbb{F}_p)[e_2]$, $R \in E(\mathbb{F}_p) \setminus \{\mathcal{O}\}$ and $e_2 - 1 = \sum_{i=0}^{\ell} n_i 2^i$ with $n_i \in \{-1, 0, 1\}$, $n_\ell = 1$

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: $N_1 \leftarrow x_R - x_P$, $D_1 \leftarrow 1$, $N_2 \leftarrow \hat{x}_R - x_P$, $D_2 \leftarrow 1$, $d_1 \leftarrow N_1$, $d_2 \leftarrow N_2$,
   $d_3 \leftarrow y_R - y_P$, $i \leftarrow \ell - 1$
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:        $t_0 \leftarrow x_R - \texttt{Tab}[k]$, $t_1 \leftarrow \hat{x}_R - \texttt{Tab}[k]$, $U_0 \leftarrow (x_R + \texttt{Tab}[k+2]) \times t_0$
21:        $U_1 \leftarrow (\hat{x}_R + \texttt{Tab}[k+2]) \times t_1$, $U_2 \leftarrow (y_R - \texttt{Tab}[k+1]) \times \texttt{Tab}[k+3]$
22:        $t_2 \leftarrow (U_0 - U_2) \bmod p$, $t_3 \leftarrow (U_1 - U_2) \bmod p$, $N_1 \leftarrow N_1^2 \cdot t_2$, $N_2 \leftarrow N_2^2 \cdot t_3$
23:        $D_1 \leftarrow (D_1 \cdot t_0)^2$, $D_2 \leftarrow (D_2 \cdot t_1)^2$, $i \leftarrow i - 1$, $k \leftarrow k + 4$
24:    elif $n_i = -1$ then
25:        $t_0 \leftarrow x_R - \texttt{Tab}[k]$, $t_1 \leftarrow \hat{x}_R - \texttt{Tab}[k]$, $U_0 \leftarrow (x_R + \texttt{Tab}[k+2]) \times t_0$
26:        $U_1 \leftarrow (\hat{x}_R + \texttt{Tab}[k+2]) \times t_1$, $U_2 \leftarrow (y_R - \texttt{Tab}[k+1]) \times \texttt{Tab}[k+3]$
27:        $t_2 \leftarrow (U_0 - U_2) \bmod p$, $t_3 \leftarrow (U_1 - U_2) \bmod p$, $N_1 \leftarrow N_1^2 \cdot t_2$, $N_2 \leftarrow N_2^2 \cdot t_3$
28:        $D_1 \leftarrow (D_1 \cdot t_0)^2 \cdot d_1$, $D_2 \leftarrow (D_2 \cdot t_1)^2 \cdot d_2$, $i \leftarrow i - 1$, $k \leftarrow k + 4$
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 The new $\mathbb{G}_1$ membership testing.

```
Input: $R \in E(\mathbb{F}_p) \setminus \{\mathcal{O}\}$
Output: if $R \in \mathbb{G}_1$, then 1; otherwise, 0
 1: Computing $N_1$, $D_1$, $N_2$, $D_2$ by Algorithm 2 or 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:     $f_1 \leftarrow (h \cdot N_1 \cdot D_2)^{\exp_1}$
 7:     $f_2 \leftarrow (h \cdot N_2 \cdot D_1)^{\exp_2}$
 8:     if $f_1 = f_2 = 1$ then
 9:         return 1
10:     else
11:         return 0
12:     end if
13: end if
```

Otherwise, we continue to performing the final exponentiation part. In this phase, we first use the trick of Montgomery's simultaneous inversion [38] to compute $f_1 = f_{e_2,P}(R)$ and $f_2 = f_{e_2,P}(\hat{\phi}(R))$ such that one inversion operation in $\mathbb{F}_p$ can be saved:

$$f_1 = \frac{N_1 \cdot D_2}{D_1 \cdot D_2}, \qquad f_2 = \frac{N_2 \cdot D_1}{D_1 \cdot D_2}. \tag{5}$$

Then, the two Tate pairings can be computed by raising f_1 and f_2 to the power of $\exp_1 = (p-1)/e_2$ and $\exp_2 = \exp_1 \cdot \tilde{m}$, respectively. If $\gcd(e_1, m) = 1$, we select $\tilde{m} = m$, i.e., $\exp_2 = (p-1)/e_1$; otherwise, we select $\tilde{m} = 1$, i.e., $\exp_2 = \exp_1$. In the first case, the exponent $\exp_2$ can be parameterized (in accordance with Table 1) by a polynomial in z with small integral coefficients. For example, the exponent

$$\exp_2 = |z^5 - z^4 - z^3 + z^2 + z + 2|$$

is attributed to BLS12-381. We observe that the given favorable case takes place for all the target curves except for BLS24-315. In Table 5, we present the exponent $\exp_2$ for the seven pairing-friendly curves. Since it is sufficient to determine whether the pairing values are equal to 1 or not, the second final exponentiation can be replaced by checking that

$$\begin{cases} f_2^{z^5+z^2+z+2} = f_2^{z^4+z^3} & \text{if} \quad z > 0, \\ f_2^{-z^5+z^4-z} = f_2^{-z^3+z^2+2} & \text{if} \quad z < 0. \end{cases}$$

In turn, the coefficients of the exponent $\exp_1$ (equal to $\exp_2$ when $\tilde{m} = 1$) in base z are not small. Indeed, the quotient polynomial $(p-1)/e_2 \in \mathbb{Q}[z]$ does not lie in the subring $\mathbb{Z}[z]$ because of $1/m$. Thus, it appears more efficient to exponentiate directly.

4.4 Computational cost

Notations. We write $\mathbf{e}_i$ and $\mathbf{z}$ for the costs of the exponentiations in $\mathbb{F}_p$ by $\exp_i$ and $|z|$. Besides, let $\mathbf{n}_1$, $\mathbf{n}_2$, $\mathbf{n}_3$, $\mathbf{n}_4$, $\mathbf{n}_5$, $\mathbf{n}_6$ and $\mathbf{n}_7$ respectively denote the numbers of the subroutines `SDBL`, `SADD`, `SSUB`, `SDADD`, `SDSUB`, $\texttt{SDSUB}_L$ and `SQPL` in the execution of the shared Miller loop.

In Table 6, we exhibit the operation counts of these subroutines. In terms of the final exponentiation part,

1. we can get an accurate estimate of $\mathbf{e}_2$ for the exponentiation by $\exp_2$ with $\tilde{m} = m$ once the cost $\mathbf{z}$ is determined (cf. Table 5);
2. we utilize the sliding window method to perform the exponentiation by $\exp_1$ (and by $\exp_2$ when $\tilde{m} = 1$), which approximately costs

$$\mathbf{e}_1 = \underbrace{\lceil \log_2(\exp_1) \rceil \Big(\frac{1}{w+1} \cdot \mathbf{m} + \mathbf{s} \Big)}_{\text{the main part of the exponentiation}} + \underbrace{(2^{w-1} - 1) \cdot \mathbf{m} + \mathbf{s}}_{\text{the precomputation}},$$

where w is the selected window size. Our experience results show that $w = 4$ is optimal for the curves under consideration.

In summary, the total cost of Algorithm 5 is about

$$\underbrace{\mathbf{n}_1 \cdot \texttt{SDBL} + \mathbf{n}_2 \cdot \texttt{SADD} + \mathbf{n}_3 \cdot \texttt{SSUB} + \mathbf{n}_4 \cdot \texttt{SDADD} + \mathbf{n}_5 \cdot \texttt{SDSUB} + \mathbf{n}_6 \cdot \texttt{SDSUB}_L + \mathbf{n}_7 \cdot \texttt{SQPL}}_{\text{the main part of the shared Miller loop}}$$
$$+ \underbrace{\mathbf{i} + 5\mathbf{m} + 3\mathbf{a}}_{\text{Eqs. (4) and (5)}} + \mathbf{e}_1 + \mathbf{e}_2. \tag{6}$$

So, the overall costs of the new $\mathbb{G}_1$ membership testing on the seven pairing-friendly curves are demonstrated in Table 7.

Table 5: The exponent $\exp_2$ for the candidate pairing-friendly curves.

curve	$\exp_2$	$\mathbf{e}_2$
BW13-310	$\lvert z^{26} - z^{13} - 3z^{12} - 3z^{11} + 3z^9 + 3z^8 - 3z^6$ $-3z^5 + 3z^3 + 3z^2 - 2\rvert$	$26\mathbf{z} + 11\mathbf{m} + 3\mathbf{s}$
BW19-286	$\lvert z^{38} - z^{19} - 3z^{18} - 3z^{17} + 3z^{15} + 3z^{14} - 3z^{12}$ $-3z^{11} + 3z^9 + 3z^8 - 3z^6 - 3z^5 + 3z^3 + 3z^2 - 2\rvert$	$38\mathbf{z} + 15\mathbf{m} + 3\mathbf{s}$
BLS12-381	$\lvert z^5 - z^4 - z^3 + z^2 + z + 2\rvert$	$5\mathbf{z} + 4\mathbf{m} + \mathbf{s}$
BLS12-446	$\lvert z^5 - z^4 - z^3 + z^2 + z + 2\rvert$	$5\mathbf{z} + 4\mathbf{m} + \mathbf{s}$
BLS24-315	$\lvert z^9 - z^8 - z^5 + z^4 + z + 2\rvert/3$	$\mathbf{e}_1$
BLS24-509	$\lvert z^9 - z^8 - z^5 + z^4 + z + 2\rvert$	$9\mathbf{z} + 4\mathbf{m} + \mathbf{s}$
BLS48-575	$\lvert z^{17} - z^{16} - z^9 + z^8 + z + 2\rvert$	$17\mathbf{z} + 4\mathbf{m} + \mathbf{s}$

Table 6: The costs of the introduced subroutines.

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	$16\mathbf{m} + 8\mathbf{m}_u + 10\mathbf{s} + \mathbf{s}_u + 26\mathbf{a} + 6\mathbf{r}$	$4\mathbf{m} + 3\mathbf{m}_u + 4\mathbf{s} + 9\mathbf{a} + 2\mathbf{r}$
SDSUB	$18\mathbf{m} + 8\mathbf{m}_u + 10\mathbf{s} + \mathbf{s}_u + 26\mathbf{a} + 6\mathbf{r}$	$6\mathbf{m} + 3\mathbf{m}_u + 4\mathbf{s} + 9\mathbf{a} + 2\mathbf{r}$
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 7: The costs of the new $\mathbb{G}_1$ membership testing for the candidate pairing-friendly curves.

curve	$(\mathbf{n}_1, \mathbf{n}_2, \cdots, \mathbf{n}_7)$	$\mathbf{z}$	without precomputation	with precomputation
BLS12-381	$(0, 2, 0, 3, 0, 0, 30)$	$5\mathbf{m} + 63\mathbf{s}$	$\mathbf{i} + 602\mathbf{m} + 244\mathbf{m}_u + 1119\mathbf{s} + 63\mathbf{s}_u + 951\mathbf{a} + 204\mathbf{r}$	$\mathbf{i} + 368\mathbf{m} + 9\mathbf{m}_u + 885\mathbf{s} + 248\mathbf{a} + 6\mathbf{r}$
BLS12-446	$(0, 2, 0, 4, 0, 0, 35)$	$6\mathbf{m} + 74\mathbf{s}$	$\mathbf{i} + 704\mathbf{m} + 287\mathbf{m}_u + 1314\mathbf{s} + 74\mathbf{s}_u + 1117\mathbf{a} + 240\mathbf{r}$	$\mathbf{i} + 428\mathbf{m} + 12\mathbf{m}_u + 1039\mathbf{s} + 292\mathbf{a} + 8\mathbf{r}$
BW13-310	$(0, 2, 2, 0, 2, 0, 10)$	$3\mathbf{m} + 11\mathbf{s}$	$\mathbf{i} + 392\mathbf{m} + 106\mathbf{m}_u + 761\mathbf{s} + 22\mathbf{s}_u + 393\mathbf{a} + 84\mathbf{r}$	$\mathbf{i} + 274\mathbf{m} + 6\mathbf{m}_u + 665\mathbf{s} + 107\mathbf{a} + 4\mathbf{r}$
BW19-286	$(1, 2, 3, 1, 0, 0, 6)$	$2\mathbf{m} + 7\mathbf{s}$	$\mathbf{i} + 340\mathbf{m} + 79\mathbf{m}_u + 667\mathbf{s} + 14\mathbf{s}_u + 284\mathbf{a} + 60\mathbf{r}$	$\mathbf{i} + 245\mathbf{m} + 3\mathbf{m}_u + 597\mathbf{s} + 79\mathbf{a} + 2\mathbf{r}$
BLS24-315	$(1, 1, 1, 1, 0, 1, 14)$	$-$	$\mathbf{i} + 389\mathbf{m} + 120\mathbf{m}_u + 806\mathbf{s} + 30\mathbf{s}_u + 467\mathbf{a} + 99\mathbf{r}$	$\mathbf{i} + 267\mathbf{m} + 3\mathbf{m}_u + 690\mathbf{s} + 125\mathbf{a} + 2\mathbf{r}$
BLS24-509	$(0, 0, 0, 2, 1, 0, 24)$	$9\mathbf{m} + 50\mathbf{s}$	$\mathbf{i} + 574\mathbf{m} + 192\mathbf{m}_u + 1299\mathbf{s} + 51\mathbf{s}_u + 753\mathbf{a} + 162\mathbf{r}$	$\mathbf{i} + 394\mathbf{m} + 9\mathbf{m}_u + 1113\mathbf{s} + 198\mathbf{a} + 6\mathbf{r}$
BLS48-575	$(1, 0, 3, 0, 1, 0, 15)$	$9\mathbf{m} + 31\mathbf{s}$	$\mathbf{i} + 558\mathbf{m} + 132\mathbf{m}_u + 1326\mathbf{s} + 32\mathbf{s}_u + 506\mathbf{a} + 108\mathbf{r}$	$\mathbf{i} + 427\mathbf{m} + 3\mathbf{m}_u + 1199\mathbf{s} + 134\mathbf{a} + 2\mathbf{r}$

5 Implementation results

We first present a Magma code to verify the correctness of our proposed formulas and algorithms. In order to compare the performance between our technique and the previous state of the art, we also provide high-speed software implementation within the RELIC toolkit. It is a well-known cryptographic library created mainly in the C programming language with ASM acceleration for the prime field arithmetic. The library contains the optimal implementations of various operations on many pairing-friendly curves, including all the BLS curves listed in Table

2. Recently, Dai et al. [15] also used RELIC to implement operations on BW13-310. Thus, we integrated our code into RELIC to ensure fair speed measurement. Our code is available at https://github.com/eccdaiy39/test-tate. All 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 compiler used was GCC version 11.4.0, with optimization `flags-O3-funroll-loops-march=native -mtune=native`.

In Figure 1, we collect the timing results (measured in $\times 10^3$ clock cycles) of each building block for the two Tate pairing evaluations on our target curves. In Figure 2, we sum these timings to compare the running time of $\mathbb{G}_1$ membership testing between our work and the previous leading work. The results show that our method without precomputation is about 62.0%, 6.5%, 2.7% and 41.2% faster than the previous optimal one on the BW13-310, BLS24-315, BLS24-509 and BLS48-575 curves, respectively. However, the former is around 50.5% and 48.6% slower than the latter on the BLS12-381 and BLS12-446 curves. Nevertheless, the performance advantage of the new method can be further extended with precomputation. In this case, it is about 2.7%, 5.3%, 110.6%, 41.4%, 40.2% and 74.4% better 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 latter cannot be sped up in advance.

To summarize, our method is well-suited for curves with a small value of ρ, such as BW13-310 and BLS48-575. Indeed, by the fact that $p \approx \#E(\mathbb{F}_p) = e_1 e_2 r$ and $e_1 \approx e_2$ for our chosen curves, it is easy to deduce that

$$\frac{\log_2(e_2)}{\log_2(r)/2} \approx \rho - 1 \gtrsim 0.$$

Recall that the previous state-of-the-art method requires approximately $\log_2(r)/2$ group operations, while the new one involves around $\log_2(e_2)$ Miller's iterations and two exponentiations in $\mathbb{F}_p$. Thus, the value of $\rho - 1$ roughly represents the ratio of the computational costs between the two methods.

Figure. 1: The timings for each building block of the two Tate pairings on the listed pairing-friendly curves.

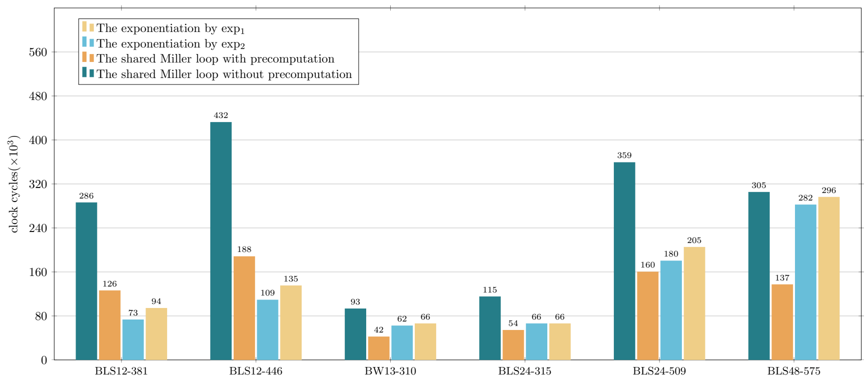
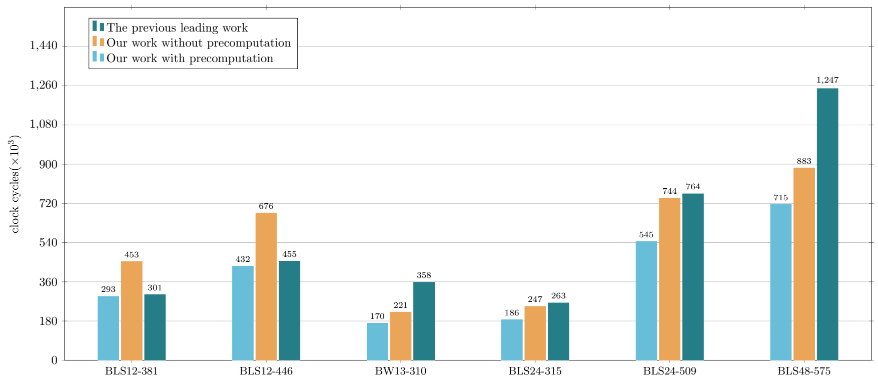


Figure. 2: The timings for $\mathbb{G}_1$ membership testing on the listed pairing-friendly curves between our work and the previous leading work.



6 Conclusion

In this paper, we revisited the problem of subgroup membership testing for $\mathbb{G}_1$ on pairing-friendly curves via the Tate pairing. We first introduced faster formulas (tailored to the BLS, BW13 and BW19 families) such that the evaluation of the two Tate pairings cumulatively requires around $2\times$ fewer Miller's iterations than the original Koshelev method. Moreover, we also provided a high-performance software implementation for our proposed algorithm and compared it to the previous leading one across several popular pairing-friendly curves. Our experimental results show that the new Koshelev-type method exhibits on BW13-310 and BLS48-575 a significant acceleration over the Scott(-type) method. With precomputation, the former is also the best for BLS24-315 and BLS24-509. However, the profit of pairing-based testing becomes negligible when applied to curves with a slightly larger value of ρ, such as BLS12-381 and BLS12-446 (not to mention the scenario $\rho \gtrsim 1.5$).

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. IACR Communications in Cryptology **1**(3) (2024). https://doi.org/10.62056/angyl86bm
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: ACM SIGSAC Conference on Computer and Communications Security – CCS 2023. pp. 3228–3238. Association for Computing Machinery, New York (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. Lecture Notes in Computer Science, vol. 9453, pp. 31–55. Springer, 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. Lecture Notes in Computer Science, vol. 9230, pp. 245–265. Springer, 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. Lecture Notes in Computer Science, vol. 2442, pp. 354–369. Springer, 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. Lecture Notes in Computer Science, vol. 3006, pp. 17–25. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24654-1_2
9. Boxall, J., El Mrabet, N., Laguillaumie, F., Le, D.P.: A variant of Miller's formula and algorithm. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing-Based Cryptography – Pairing 2010. Lecture Notes in Computer Science, vol. 6487, pp. 417–434. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17455-1_26
10. 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
11. 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 – CANS 2020. Lecture Notes in Computer Science, vol. 12579, pp. 280–298. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-65411-5_14
12. Costello, C., Stebila, D.: Fixed argument pairings. In: Abdalla, M., Barreto, P.S.L.M. (eds.) Progress in Cryptology – LATINCRYPT 2010. Lecture Notes in Computer Science, vol. 6212, pp. 92–108. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14712-8_6
13. Dai, Y., He, D., Peng, C., Yang, Z., Zhao, C.A.: Revisiting pairing-friendly curves with embedding degrees 10 and 14. In: Chung, K.M., Sasaki, Y. (eds.) Advances in Cryptology – ASIACRYPT 2024. Lecture Notes in Computer Science, vol. 15485, pp. 454–485. Springer, Singapore (2025). https://doi.org/10.1007/978-981-96-0888-1_15
14. 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
15. 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
16. 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

17. Eisenträger, K., Lauter, K., Montgomery, P.L.: Fast elliptic curve arithmetic and improved Weil pairing evaluation. In: Joye, M. (ed.) Topics in Cryptology – CT-RSA 2003. Lecture Notes in Computer Science, vol. 2612, pp. 343–354. Springer, Berlin, Heidelberg (2003). https://doi.org/10.1007/3-540-36563-X_24
18. 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. Lecture Notes in Computer Science, vol. 13276, pp. 367–396. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-07085-3_13
19. 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. Lecture Notes in Computer Science, vol. 13503, pp. 518–536. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17433-9_22
20. Electric Coin Company: What is Jubjub?, `https://bitzecbzc.github.io/technology/jubjub`
21. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Journal of Cryptology **23**(2), 224–280 (2010). https://doi.org/10.1007/s00145-009-9048-z
22. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge (2019), `https://eprint.iacr.org/2019/953`
23. 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. Lecture Notes in Computer Science, vol. 2139, pp. 190–200. Springer, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_11
24. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016. Lecture Notes in Computer Science, vol. 9666, pp. 305–326. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_11
25. 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. Lecture Notes in Computer Science, vol. 12111, pp. 535–564. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_19
26. Guillevic, A.: Pairing-friendly curves (2021), `https://members.loria.fr/AGuillevic/pairing-friendly-curves`
27. Hamburg, M.: Ed448-Goldilocks, a new elliptic curve (2015), `https://eprint.iacr.org/2015/625`
28. Hamburg, M.: Computing the Jacobi symbol using Bernstein–Yang (2021), `https://eprint.iacr.org/2021/1271`
29. 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. Lecture Notes in Computer Science, vol. 8365, pp. 45–61. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04873-4_3
30. 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
31. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) Advances in Cryptology – ASIACRYPT 2010. Lecture Notes in Computer Science, vol. 6477, pp. 177–194.

Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_11
32. 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. Lecture Notes in Computer Science, vol. 9814, pp. 543–571. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_20
33. Koshelev, D.: Subgroup membership testing on elliptic curves via the Tate pairing. Journal of Cryptographic Engineering **13**(1), 125–128 (2023). https://doi.org/10.1007/s13389-022-00296-9
34. Koshelev, D.: Correction to: Subgroup membership testing on elliptic curves via the Tate pairing. Journal of Cryptographic Engineering **14**(1), 127–128 (2024). https://doi.org/10.1007/s13389-023-00331-3
35. Masson, S., Sanso, A., Zhang, Z.: Bandersnatch: a fast elliptic curve built over the BLS12-381 scalar field. Designs, Codes and Cryptography **92**(12), 4131–4143 (2024). https://doi.org/10.1007/s10623-024-01472-0
36. Mbiang, N.B., Aranha, 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
37. 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
38. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation **48**(177), 243–264 (1987). https://doi.org/10.1090/S0025-5718-1987-0866113-7
39. 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. Lecture Notes in Computer Science, vol. 11922, pp. 243–272. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34621-8_9
40. 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
41. Scott, M.: MIRACL: multiprecision integer and rational arithmetic C/C++ library, `https://github.com/miracl/core`
42. Scott, M.: A note on group membership tests for $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ on BLS pairing-friendly curves (2021), `https://eprint.iacr.org/2021/1130`
43. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory **56**(1), 455–461 (2009). https://doi.org/10.1109/TIT.2009.2034881
44. Washington, L.C.: Elliptic curves: number theory and cryptography, Discrete Mathematics and Its Applications, vol. 50. CRC Press, Boca Raton, 2 edn. (2008). https://doi.org/10.1201/9781420071474

A Formulas for `SDBL`, `SSUB` and `SDSUB`

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

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

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

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

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 $g_{2i,P}$ can be obtained from $g_{i,P}$ as follows:

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

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

$$N_{2i}(x, y) = N_i^2(x, y) \cdot Z_{2T} \cdot (xZ_{2T}^2 - X_{2T}),$$
$$D_{2i}(x, y) = D_i^2(x, y) \cdot \left(yZ_{2T}^3 - Y_{2T} + \frac{3}{2}X_T^2 \cdot (xZ_{2T}^2 - X_{2T})\right).$$

We first compute the point $2T = (X_{2T}, Y_{2T}, Z_{2T})$. Then, the tuple $\tau_{2i}(R)$ can be obtained at a cost of $9\mathbf{m} + 3\mathbf{m}_u + 5\mathbf{s} + 8\mathbf{a} + 2\mathbf{r}$ by performing the sequence of operations

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

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 $g_{i-1,P}$ can be obtained from $g_{i,P}$ as follows:

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

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

$$N_{i-1}(x,y) = N_i(x,y)\cdot(xZ_{T-P}^2 - X_{T-P}),$$
$$D_{i-1}(x,y)= D_i(x,y)\cdot Z_{T-P}\cdot\big((y-y_P)\cdot Z_{T-P} + \theta_{T-P}\cdot(x-x_P)\big),$$

where $\theta_{T-P} = -y_P\cdot Z_T^3 - Y_T$ can be obtained during the computation of $T-P$. Thus, the tuple $\tau_{i-1}(R)$ can be found at a cost of $8\mathbf{m}+3\mathbf{m}_u+\mathbf{s}+6\mathbf{a}+2\mathbf{r}$ via the sequence of operations

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

where d_1, d_2 and d_3 are given at the initial stage of the shared Miller loop (Line 1 in Algorithms 2 and 4). 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 $g_{i,P}$ and $g_{2i-1,P}$ satisfy the relation

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

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

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

where

$$\beta_{2T-P} = x_P\cdot Z_{2T}^2 - X_{2T}, \qquad \theta_{2T-P} = -y_P\cdot Z_{2T}^3 - Y_{2T}.$$

In order to compute $\tau_{2i-1}(R)$, we first determine

$$2T = (X_{2T}, Y_{2T}, Z_{2T}), \qquad 2T-P = (X_{2T-P}, Y_{2T-P}, Z_{2T-P}).$$

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 find the above tuple by performing the sequence of operations

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

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 $18\mathbf{m}+8\mathbf{m}_u+10\mathbf{s}+\mathbf{s}_u+26\mathbf{a}+6\mathbf{r}$.