[1]

Faster Beta Weil Pairing on BLS Pairing Friendly Curves with Odd Embedding Degree

Azebaze Guimagang Laurian, Fouotsa Emmanuel, El Mrabet Nadia and Pecha Njiahouo Aminatou

Abstract. Since the advent of pairing-based cryptography, various optimization methods that increase the speed of pairing computations have been exploited, as well as new types of pairings. This paper extends the work of Kinoshita and Suzuki who proposed a new formula for the β-Weil pairing on curves with even embedding degree by eliminating denominators and exponents during the computation of the Weil pairing. We provide novel formulas suitable for the parallel computation for the β-Weil pairing on curves with odd embedding degree which involve vertical line functions useful for sparse multiplications. For computations we used Miller's algorithm combined with storage and multifunction methods. Applying our framework to BLS-27, BLS-15 and BLS-9 curves at respectively the 256 bit, the 192 bit and the 128 bit security level, we obtain faster β-Weil pairings than the previous state-of-the-art constructions. The correctness of all the formulas and bilinearity of pairings obtained in this work is verified by a SageMath code.

Mathematics Subject Classification (2010). Primary 14H52; Secondary 1990S.

Keywords. β-Weil pairing and Optimal Ate pairing and Multifunction technique and Storage technique and BLS curves.

1. Introduction

Pairings have first been studied in mathematical research areas such as algebraic number theory and algebraic geometry [2]. Then pairings on elliptic curves have attracted significant attention and have been applied to many cryptographic schemes, namely Boneh and Franklin's identity based encryption scheme [3], Boneh, Lynn and Shacham's short signature scheme [4] and Joux's one round tripartite key exchange [5]. Before the advent of pairings

[1] An extended and improved version of this paper will be published in Mathematics and Computer Science

Authors 1, 2 and 4 are supported by the Simons Foundation through the project PREMA, Subsaharan Africa. The second author aknowledges the support of TWAS UNESCO under the Grant 20-063 RG/MATHS/AF/AC-I.

one did not know how to effectively build up these protocols. A pairing is a bilinear map from the cartesian product of two abelian additive groups to an abelian multiplicative group. For instance, the Weil pairing e over the r-torsion $E[r]$ of an elliptic curve E, satisfies the following basic properties :

- e is bilinear i.e. for every P_1, P_2, Q_1 and Q_2 in $E[r]$, we have that $e(P_1 + P_2, Q_1) = e(P_1, Q_1) \cdot e(P_2, Q_1)$ and $e(P_1, Q_1 + Q_2) = e(P_1, Q_1) \cdot e(P_1, Q_2)$;
- e is non degenerate i.e. if $e(P, Q) = 1$ for all $P \in E[r]$, then $Q = \mathcal{O}$, where $\mathcal{O}$ is the point at infinity;
- e is alternating i.e. $e(P, P) = 1$ for all P in $E[r]$.

The bilinearity is the most important property for building applications with cryptographic pairings. The very first pairings used in cryptography were the Weil and the Tate pairings [6, 3]. To date there are several variants of the Tate pairing and the most efficient candidate is the optimal Ate pairing [7]. On the other hand variants of the Weil pairing such as α-Weil [8], β-Weil [8, 9, 10] and ω-Weil pairings [11, 12] are elegant constructions tailored for parallel executions. Viewed in this angle, the β-Weil pairing is more efficient than the optimal Ate pairing [9, 10]. Moreover our motivation for computing the β-Weil pairing rather than the Tate pairing or its variants, is that it has no time-consuming final exponentiation and can be trivially parallelized [9]. The pairings can be implemented on a smart card, but due to the limited computing power and constrained memory there is a need to faster evaluate the pairings.

In this work, we focus on the β-Weil pairing defined over pairing-friendly elliptic curves with odd embedding degrees $9, 15$ and 27. These curves admit twists of degree three which enable computations to be done in subfields and also lead to the denominator elimination technique. It is noticed in [13] that elliptic curves with embedding degree $k = 27$ are a suitable choice for computing product of pairings. Moreover, Barbulescu et *al.* showed that at the 256 bits security level, curve with $k = 27$ seem to be the best choice for pairing efficiency [14]. Also in [10] Fouotsa et *al.* proposed a pairing denoted $\hat{\beta}_k$-pairing different from β-pairing which is more efficient than the optimal Ate pairing over BLS-15. Many works have been carried out for faster evaluation of the pairings in the literature such as [15, 16, 17, 18, 19] including their security see [20, 21, 22].

The original β-Weil pairing that we consider here is the generalised formula proposed by Fouotsa et *al.* in [10]. Their theoretical evaluation showed some limitations such as:
− The inversions of Miller's functions which are very costly,
− The precomputation of some points $[p^i]P$ that underestimated the total cost of the β-Weil pairing,
− the β-Weil pairing was not tailored for multifunction technique, this technique extremely reduces the cost of squarings in the finite field.

Our main contribution is the extension of Kinoshita and Suzuki's work [1] who proposed a formula for the β-Weil pairing on the curves with even embedding degree by eliminating the denominators and the exponents. But

their formula to simplify the denominators does not work for odd embedding degrees. We then provide a novel formula for the β-Weil pairing on curves with odd embedding degrees which involve vertical line functions useful for sparse multiplications. To compute the new formula of our novel β-Weil pairing, we use Miller's algorithm based on storage and multifunction techniques.

Whereas, storage technique consists of compute and store line functions for Miller's loop that are reused to find another line functions for other Miller's function, multifunction technique evaluates the product of n Miller's functions and only requires a single squaring in the extension field per iteration instead of n squarings in the naive way. The idea of precomputation and storage of line functions and the exploitation of sparsity can be found in [23] and for multi-function technique see [24].

The correctness of the formulas for curves with embedding degrees $27, 15$ and 9 are ensured by a SageMath script available at [25] and inspired by Aurore's online SageMath repository for even embedding degree pairings. As applications we applied the variant of Miller's algorithm with these formulas with the Miller loop parameters obtained by applying the recommendation in [26] and [27, 28]. The pairings over these curves are resistant to the STNFS attacks.

Detailed cost estimation of β-Weil pairing on BLS-27, BLS-15 and BLS-9 at respectively the $256, 192$ and 128 bits level of security are provided. Our theoretical results reduce the number of multiplication operations on the prime field in β-Weil pairing for BLS family with $k = 27, 15$ and 9 about 44.78%, 49.07% and 38.49%, as well as the number of divisions of about 87.1% 78.8% and 61.2% respectively for serial computation. The optimal Ate pairing remains the fastest pairing in the serial computation. However, the proposed β-Weil pairing on curve with $k = 15$ is competitive to optimal Ate pairing on the same curve. In the parallel computation with 3 processors, the β-Weil pairing on BLS-27, BLS-15 and BLS-9 is faster than the optimal Ate pairing. For more details see Section 6, Appendix A and Table 3.

This paper is organized as follows. The Section 2 summarises the mathematical background on Weil pairing over elliptic curves and recalls the idea of Kinoshita and Suzuki [1] to accelerate the β-Weil pairing computation. Section 3 provides a new formula of the β-Weil pairing on curves with odd embedding degrees. Section 4 describes storage and multifunction techniques. Section 5 estimates the theoretical cost of the basic and special operations of the pairing. Sections 6 provides theoretical costs of computing, the new β-Weil pairing both in serial and parallel and section 7 compares our results with those done on previous works. Finally, Section 8 concludes the work.

2. Mathematical Preliminaries.

In this section, we define the Weil pairing as well as its variant called β-Weil pairing with some past results.

2.1. The Weil pairing

Let E be an elliptic curve defined over $\mathbb{F}_p$, where p is a large prime number and let r be the largest prime number such that r divides $\#E(\mathbb{F}_p)$. Let k be the smallest positive integer such that r divides p^k-1. The integer k is called the embedding degree of E (with respect to r). For any $P \in E[r]$, we denote $f_{r,P}$ the rational function with divisor

$$div(f_{r,P}) = r[P] - r[\mathcal{O}].$$

The Weil pairing is defined as:

$$e_W : E(\mathbb{F}_{p^k})[r] \times E(\mathbb{F}_{p^k})[r] \longrightarrow \mu_r : (P,Q) \longmapsto (-1)^r \frac{f_{r,P}(Q)}{f_{r,Q}(P)}.$$

The function $f_{r,P}$ is called the Miller function, it is obtained through Miller's algorithm [17] based on the following relations. For all $a, b \in \mathbb{Z}$ and $P \in E[r]$,

$$f_{1,P} = 1, \tag{2.1}$$

$$f_{a+b,P} = f_{a,P} \cdot f_{b,P} \cdot h_{[a]P,[b]P}, \tag{2.2}$$

$$f_{ab,P} = f_{b,P}^a \cdot f_{a,[b]P} = f_{a,P}^b \cdot f_{b,[a]P}, \tag{2.3}$$

where $h_{[a]P,[b]P} = l_{[a]P,[b]P}/\mathcal{V}_{[a+b]P}$ and $l_{[a]P,[b]P}$ is the straight line containing $[a]P$ and $[b]P$ and $\mathcal{V}_{[a+b]P}$ is the corresponding vertical line passing through $[a+b]P$.

The Weil pairing is composed of two executions of the Miller's loop for the evaluation of $f_{r,P}(Q)$ and $f_{r,Q}(P)$. The efficiency of Miller's loop highly depends on the choice of the elliptic curve and the system of coordinates of the elliptic curve. Pairing-friendly elliptic curves are curves allowing an efficient implementation of pairings. These curves are parametrized by polynomials $p(x), r(x)$ and $t(x)$ such that for a given security level, we find the corresponding value of x such that $\#E(\mathbb{F}_{p(x)}) = p(x) + 1 - t(x)$ is divisible by $r(x)$. The parameters $p(x), r(x)$ and $t(x)$ for elliptic curves with embedding degrees $9, 15$ and 27 denoted as BLS-9, BLS-15 and BLS-27 in [29] are respectively represented as polynomials as follows:
BLS-27: $r(x) = \frac{1}{3}(x^{18}+x^9+1)$, $p(x) = \frac{1}{3}(x-1)^2(x^{18}+x^9+1)+x$, $t(x) = x+1$.
BLS-15: $r(x) = x^8 - x^7 + x^5 - x^4 + x^3 - x + 1$, $p(x) = \frac{1}{3}(x^{12} - 2x^{11} + x^{10} + x^7 - 2x^6 + x^5 + x^2 + x + 1)$, $t(x) = x + 1$.
BLS-9: $r(x) = \frac{1}{3}(x^6 + x^3 + 1)$, $p(x) = \frac{1}{4}((x+1)^2 + \frac{1}{3}((x-1)^2(2x^3+1)^2))$, $t(x) = x + 1$.

2.2. Previous results on β-Weil pairing computation

The most efficient way to optimize a pairing is by using Miller's algorithm which computes the rational function $f_{s,R}$ where, $s \in \mathbb{Z}$ and $R \in E[r]$. This rational function is called Miller's function with divisor $div(f_{s,R}) = s(R) - ([s]R) - (s-1)(\mathcal{O})$. Vercauteren [7] introduced the extended Miller function $f_{s,h,R}$ with divisor $div(f_{s,h,R}) = \sum_{i=0}^{w} h_i(([p^i]R) - (\mathcal{O}))$, where $h(x) = \sum_{i=0}^{w} h_i x^i$ in $\mathbb{Z}[x]$ is the optimal polynomial obtained by using a

lattice-based method such that $h(p) = 0 \mod r$. For $m = h(p)/r$ and $m \nmid r$, this extended Miller function is expressed as

$$f_{s,h,R} = \frac{f_{r,R}^{m}}{\prod_{j=0}^{w} f_{p^j,R}^{h_j}}. \tag{2.4}$$

The extended Miller's function was used to propose efficient variants of Ate pairing and Weil pairing such as optimal Ate pairing [7] and β-Weil pairing [8] on curves of even embedding degrees where the Miller's loop length is reduced to $\log_2(x)$ iterations.

In [10], Fouotsa et *al.* extended the work of Aranha et *al.* [8] on the β-Weil pairing to curves with an odd embedding degree k. They provided the following formula.

Theorem 2.1 ([10]**, Theorem 4).** *Let l be a proper divisor of k. There exists a polynomial $h(z) = \sum_{i=0}^{w} h_i z^i \in \mathbb{Z}[z]$ such that $|h_i| < r^{\frac{1}{\varphi(k)}}$ and $h(p) = rs$ so that the map*

$$\beta_k : \mathbb{G}_1 \times \mathbb{G}_2 \to \mu_r : (P, Q) \mapsto \left(\prod_{i=0}^{e-1} \left(\frac{f_{p,h,Q}([p^i]P)}{f_{p,h,[p^i]P}(Q)} \right)^{p^{e-1-i}} \right)^{p^l - 1},$$

is a pairing and $e = k/d$ where d is the twisted degree of the curve. More precisely, if $r \nmid sp^{e-1-i}$ and $r \nmid h_j$, for all $0 \leq i \leq e-1$ and $1 \leq j \leq w$, then the map β_k is non-degenerate.

Note that the final powering $p^l - 1$ sends to 1 the multiplicative factors of the Miller's function which lies in proper subfields of $\mathbb{F}_{p^k}$.

Remark 2.2. For curves of even embedding degrees, Aranha [8] et *al.* provided the following formula of the β-Weil pairing

$$\beta_k(P, Q) = \left(\prod_{i=0}^{e-1} \left(\frac{f_{p,h,Q}([p^i]P)}{f_{p,h,[p^i]P}(Q)} \right)^{p^{e-1-i}} \right)^{p^{k/2} - 1}.$$

To avoid denominators and exponents in the computation of β-Weil pairings for curves of even embedding degrees, Kinoshita et *al.* [1] considered the useful results in the following lemma.

Lemma 2.3. 1. *Elimination of the denominators (and thus an inversion) of the β-Weil pairing. In the context of an elliptic curve of even embedding degree, the following relations hold:*

$$f_{a,R}^{-1} = f_{a,-R} \qquad \text{and} \qquad f_{p,h,R}^{-1} = f_{p,h,-R}$$

2. *Elimination of the exponents of the β-Weil pairing. For any $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$:*

$$f_{p,h,P}^{p^i}(Q) = f_{p,h,P}(\pi_{p^i}(Q)) \qquad \text{and} \qquad f_{p,h,Q}^{p^i}(P) = f_{p,h,\pi_{p^i}(Q)}(P).$$

Then they provided the simplified formula of the β-Weil pairing

$$\beta_k(P,Q) = \left(\prod_{i=0}^{e-1} f_{p,h,\pi_{p^{\delta_i}}(Q)}([p^i]P) \cdot f_{p,h,[p^i]\overline{P}}(\pi_{p^{\delta_i}}(Q))\right)^{p^{k/2}-1} \tag{2.5}$$

where, $\overline{P} = -P$ and $\delta_i = e-1-i$.

The first item of the Lemma 2.3 does not work on elliptic curves with odd embedding degrees. In the following we will derive an analogous lemma as well as the new β-Weil pairing on elliptic curves of odd embedding degrees.

3. New Formula for β-Weil pairing

In this section we first provide a new general formula of β-Weil pairing on elliptic curves of odd embedding degrees. Then we give the explicit formulas of β-Weil pairing on BLS-27, BLS-15 and BLS-9.

3.1. New formula for β-Weil pairing: the case of odd embedding degrees

The following lemma gives nice properties of the Miller's function in the case of odd embedding degrees.

Lemma 3.1. *For all $a \in \mathbb{Z}$ and $R \in E$, we obtain the following two relations:*

(i) $f_{a,R}^{-1} = f_{a,-R} \cdot \mathcal{V}_{[a]R} \cdot \mathcal{V}_R^{-a}$.
(ii) $f_{p,h,R}^{-1} = f_{p,h,-R} \cdot \prod_{j=0}^{w} \mathcal{V}_{[p^j]R}^{-h_j}$.

Proof. (i) From Eq. 2.3, We have that, $f_{a,R}^{-1} = \dfrac{f_{-1,R}^a \cdot f_{a,[-1]R}}{f_{-1,[a]R}}$.
Since, $div(f_{-1,[a]R}) = -(([a]R) + ([-a]R)) - 2(\mathcal{O}) = div(\mathcal{V}_{[a]R}^{-1})$ and $div(f_{-1,R}^a) = di(\mathcal{V}_R^{-a})$. Then, $f_{a,R}^{-1} = f_{a,-R} \cdot \mathcal{V}_{[a]R} . \mathcal{V}_R^{-a}$.

(ii) From Eq. 2.4, we have that $f_{p,h,R} = \dfrac{f_{rm,R}}{\prod_{i=1}^{w} f_{p^i,R}^{h_i}}$. Taken this equation to power -1, it yields $f_{p,h,R}^{-1} = \dfrac{f_{rm,R}^{-1}}{\prod_{i=1}^{w}(f_{p^i,R}^{-1})^{h_i}}$. Then from (i), $f_{p,h,R}^{-1} = \dfrac{f_{rm,-R} \cdot \mathcal{V}_{[rm]R} \cdot \mathcal{V}_R^{-rm}}{\prod_{i=1}^{w}\left(f_{p^i,-R}^{h_i} \cdot \mathcal{V}_{[p^i]R}^{h_i} \cdot \mathcal{V}_R^{-p^i h_i}\right)} = \dfrac{f_{rm,-R}}{\prod_{i=1}^{w} f_{p^i,-R}^{h_i}} \cdot \dfrac{\mathcal{V}_{[rm]R} \cdot \mathcal{V}_R^{-rm}}{\prod_{i=1}^{w}(\mathcal{V}_{[p^i]R}^{h_i}) \cdot \mathcal{V}_R^{-\sum_{j=0}^{w} h_i p^i}}$.
Since, $[r]R = \mathcal{O}$ and $\sum_{j=0}^{w} h_j p^j = r \cdot m$, the result follows
$f_{p,h,R}^{-1} = f_{p,h,-R} \cdot \prod_{j=0}^{w} \mathcal{V}_{[p^j]R}^{-h_j}$.

□

A straightforward application of Lemma 3.1 in the Theorem 2.1 gives the following theorem:

Theorem 3.2. *For every curves of odd embedding degrees, the new formula of β-Weil pairing is given as follows*

$$\beta_k(P,Q) = \left(\prod_{i=0}^{e-1} f_{p,h,\pi_{p^{\delta_i}}(Q)}([p^i]P) \cdot f_{p,h,[p^i]\overline{P}}(\pi_{p^{\delta_i}}(Q)) \cdot \prod_{j=0}^{w} \mathcal{V}_{[p^{i+j}]P}^{-h_j}(\pi_{p^{\delta_i}}(Q)) \right)^{p^l-1}, \tag{3.1}$$

where $\overline{P} = -P$ and $\delta_i = e - 1 - i$.

Corollary 3.3. *For pairing-friendly curves of embedding degrees $k = 27, 15$ and 9, the polynomial $h(z)$ for the extended Miller's function yields $h(z) = x - z$, then $f_{p,h,P} = f_{x,P}$ and Eq. 3.1 becomes*

$$\beta_k(P,Q) = \left(\prod_{i=0}^{e-1} f_{x,Q_i}(P_i) \cdot f_{x,\overline{P}_i}(Q_i) \cdot \mathcal{V}_{P_i}^{-x}(Q_i) \cdot \mathcal{V}_{P_{i+1}}(Q_i) \right)^{p^l-1}, \tag{3.2}$$

where $P_i = [p^i]P$ and $Q_i = \pi_{p^{\delta_i}}(Q)$.

Enge and Milan [30] proposed a variant of Miller's algorithm (Algorithm 2) which evaluates the function $f_{x,P}$ or $f_{x,Q}$ for any seed x, positive, negative, sparse, in binary and in $2-$NAF form, that is, $x = \sum_{i=0}^{n} s_i 2^i$ with s_i in $\{0, -1, 1\}$.

Moreover, for a matter of efficiency, it is possible to find the parameters x, so as to avoid the computation of $\mathcal{V}_{P_i}^{x}(Q_i)$. The simplified formulas in this case are given by corollary 3.4.

Corollary 3.4. *The simplified formulas of the β-Weil pairing with $x = -2^n + \sum_{i=0}^{n-1} s_i 2^i$, or $x = 2^n + \sum_{i=0}^{n-1} s_i 2^i$, where $s_i \in \{0, -1, 1\}$, is*

$$\beta_k(P,Q) = \left(\prod_{i=0}^{e-1} f_{x,Q_i}(P_i) \cdot f_{-x,P_i}(Q_i) \cdot \mathcal{V}_{P_{i+1}}(Q_i) \right)^{p^l-1}, \tag{3.3}$$

where $P_i = [p^i]P$ and $Q_i = \pi_{p^{\delta_i}}(Q)$.

Proof. If $x < 0$, that is $x = -|x|$ since, $\overline{P}_i = -P_i$ then, $f_{x,\overline{P}_i}(Q_i) = f_{-|x|,-P_i}(Q_i)$. From Eq. 2.3, $f_{-|x|,-P_i}(Q_i) = f_{-1,-P_i}^{|x|}(Q_i) \cdot f_{|x|,P_i}(Q_i)$. But, $f_{-1,-P_i}(Q_i) = \mathcal{V}_{-P_i}^{-1}(Q_i)$ and $\mathcal{V}_{-P_i}^{-1}(Q_i) = \mathcal{V}_{P_i}^{-1}(Q_i)$, therefore $f_{x,\overline{P}_i}(Q_i) = f_{|x|,P_i}(Q_i) \cdot \mathcal{V}_{P_i}^{-|x|}(Q_i) = f_{-x,P_i}(Q_i) \cdot \mathcal{V}_{P_i}^{x}(Q_i)$.

If $x > 0$, $f_{-x,P_i}(Q_i) = f_{x,-P_i}(Q_i) \cdot \mathcal{V}_{P_i}^{-x}(Q_i)$ i.e $f_{x,\overline{P}_i}(Q_i) = f_{-x,P_i}(Q_i) \cdot \mathcal{V}_{P_i}^{x}(Q_i)$.

For each case substitute, $f_{x,\overline{P}_i}(Q_i)$ by $f_{-x,P_i}(Q_i) \cdot \mathcal{V}_{P_i}^{x}(Q_i)$ in Eq. 3.2 and it becomes Eq. 3.3. □

Algorithm 1 computes $f_{x,Q}(P)$ for positive x, whereas Algorithm 2 computes $f_{x,Q}(P)$ for negative x. In fact, $f_{x,Q}(P) = f_{-x,-Q}(P) \cdot \mathcal{V}_{Q}^{x}(P)$ and the Algorithm 2 computes $\mathcal{V}_{Q}^{x}(P)$ internally.

Algorithm 1: Evaluate Full Miller's function $f_{x,Q}(P)$ for $x > 0$.

Input: $x = 2^n + \sum_{i=0}^{n-1} s_i 2^i$, where $s_i \in \{0, 1\}$
Output: $f_{x,Q}(P), [x]Q$

$V \leftarrow Q$,
$f \leftarrow 1$,
for *i from* $n-1$ *down to* 0 **do**
  $f \leftarrow f^2.h_{V,V}(P), \qquad V \leftarrow [2]V$
  if $s_i = 1$ **then**
    $f \leftarrow f.h_{V,Q}(P), \qquad V \leftarrow V + Q$
return f, V.

Algorithm 2: Evaluate Full Miller's function $f_{x,Q}(P)$ for $x < 0$.

Input: $x = -2^n + \sum_{i=0}^{n-1} s_i 2^i$, where $s_i \in \{0, -1\}$
Output: $f_{x,Q}(P), [x]Q$

$V \leftarrow -Q$,
$f \leftarrow \mathcal{V}_Q^{-1}(P)$,
for *i from* $n-1$ *down to* 0 **do**
  $f \leftarrow f^2.h_{V,V}(P), \qquad V \leftarrow [2]V$
  if $s_i = -1$ **then**
    $f \leftarrow f.h_{V,-Q}(P).\mathcal{V}_Q^{-1}(P), \qquad V \leftarrow V - Q$
return f, V.

By direct analogy with Algorithm 1 and 2, exchanging P and Q gives two algorithms for computing Miller Lite function this is, $f_{x,P}(Q)$ for positive x and $f_{x,P}(Q)$ for negative x respectively. Note that the Miller Lite function is defined over $\mathbb{F}_p$ and evaluated at a point of $E(\mathbb{F}_{p^k})$. Whereas the Full Miller function is defined over $\mathbb{F}_{p^k}$ and evaluated at a point of $E(\mathbb{F}_p)$.

Remark 3.5. Lin et *al.* [31] first observed that, for every point S in $\mathbb{G}_1$ and R in $\mathbb{G}_2$, we have that

$$\frac{1}{\mathcal{V}_R(S)} = \frac{1}{x_S - x_R} = \frac{x_S^2 + x_S x_R + x_R^2}{y_S^2 - y_R^2}.$$

This leads to the denominator elimination with cubic twist since, $y_S^2 - y_R^2$ lies in a subfield. Then $\mathcal{V}_R^{-1}(S)$ can be replaced by

$$\mathcal{S}_R(S) = x_S^2 + x_S x_R + x_R^2. \tag{3.4}$$

3.2. Applications on BLS curves: explicit formulas

In this subsection, we apply the proposed Formula 3.3 to BLS pairing friendly curves. BLS curves are available for several embedding degrees [29], in particular, the curves with embedding degrees $k = 27, 15$ and $k = 9$ named BLS-27, BLS-15 and BLS-9 respectively. These curves have the form $y^2 = x^3 + b$ and admit a cubic twist.

Table 1 provides explicit formulas of β-Weil pairing on BLS-27, BLS-15 and BLS-9 at different levels of security.

TABLE 1. New β-Weil pairing formulas on BLS curves. At 128-bit, 192-bit level of security the seeds provided come from the recommendation of Guillevic in [26]. At 256−bit security level no seed is given for $k = 27$ we then refer to [27].

Curves	Level of security and the x value	Size of p	the β-Weil pairing
BLS-27	256-bit $-2^{51} - 2^{31} - 2^{21} - 2^8 - 2^4$	1019	$\left(\prod_{i=0}^{8} f_{x,Q_i}(P_i) \cdot f_{-x,P_i}(Q_i) \cdot \mathcal{V}_{P_{i+1}}(Q_i)\right)^{p^9-1}$
BLS-15	192-bit $-2^{77} - 2^{76} - 2^{68} - 2^{50}$	930	$\left(\prod_{i=0}^{4} f_{x,Q_i}(P_i) \cdot f_{-x,P_i}(Q_i) \cdot \mathcal{V}_{P_{i+1}}(Q_i)\right)^{p^5-1}$
BLS-9	128-bit $-2^{74} - 2^{72} - 2^{46} - 2^{31}$	593	$\left(\prod_{i=0}^{2} f_{x,Q_i}(P_i) \cdot f_{-x,P_i}(Q_i) \cdot \mathcal{V}_{P_{i+1}}(Q_i)\right)^{p^3-1}$

These new formulas of the β-Weil pairing are more suitable for the storage and multifunction techniques as explained in the next section.

4. Storage and Multifunction Techniques For Fast Computation

The storage technique stores some line equations during the Miller's algorithm in order to speed up the computation, whereas the multifunction technique reduces the number of squarings in the pairing computation.

4.1. Storage technique

The computation of the Miller's function $f_{s,Q}(P)$ can be performed in two steps: the first step consists of evaluating the line functions depending on Q, while the second step is to evaluate the line functions at P. Storage technique then computes and stores the line functions of the first step. As Algorithm 3 computes and stores all line functions of $f_{s,Q}$ or $f_{s,P}$, Algorithm 4 computes and stores line functions of $f_{s,\pi_{p^i}(Q)}$ by raising all the line functions of $f_{s,Q}$ to the power p^i.

Algorithm 3: CSL: Compute and Store Line functions for $s < 0$

Input: $R \in \mathbb{G}_1$ (or $R \in \mathbb{G}_2$),
integer $s = -2^n + \sum_{i=0}^{n-1} s_i 2^i$, where $s_i \in \{0, -1\}$
Output: An array g of $\lfloor \log_2 s \rfloor + HW(s) - 1$ line functions and $[s]R$.
$HW(s)$ is the Hamming weight of s

1 $T \leftarrow -R$ and $j \leftarrow 1$
2 **for** $i \leftarrow n-1$ *to 0* **do**
3 $\quad g[j] \leftarrow h_{T,T}$, $T \leftarrow 2T$, $j \leftarrow j+1$
4 $\quad$ **if** $s_i \neq 0$ **then**
5 $\quad\quad g[j] \leftarrow h_{T,R}$, $T \leftarrow T - R$, $j \leftarrow j+1$
6 return g, T.

We remark that, for $s > 0$, item 1 becomes $T = R$ and item 5 is $T = T + R$.

Algorithm 4: CSLFrob: p^i-th Frobenius for Stored Line functions [1]

Input: An array g of line functions, integer i
Output: An array g' of line functions

1 **for** $j \leftarrow 1$ *to length of* g **do**
2 $\quad g'[j] \leftarrow g[j]^{p^i}$
3 return g'.

4.2. Multifunction technique

After computing and storing all line functions, we compute the products of e Miller's functions with the help of multifunction technique. The method only requires a single squaring in the extension field per doubling step instead of e squarings in the naive way. Algorithm 5 evaluates the product of e-multifunction.

Algorithm 5: EPM: Evaluate Product of e-Multifunction for $s < 0$

Input: $\left[(g_0, P_0), ..., (g_{e-1}, P_{e-1}), (h_0, Q_0), ..., (h_{e-1}, Q_{e-1})\right]$
$s = -2^n + \sum_{j=0}^{n-1} s_j 2^j$, where $s_j \in \{0, -1\}$
$h_i's$ are the stored line functions from f_{-s,P_i}
$g_i's$ are the stored line functions from f_{s,Q_i}
Output: $\prod_{i=0}^{e-1} \left(f_{s,Q_i}([p^i]P) \cdot f_{-s,P_i}(Q_i)\right)$,

$f \leftarrow \prod_{k=0}^{e-1} \mathcal{S}_{-Q_k}(P_k)$, (see Eq. 3.4)
$j \leftarrow 1$
for *i from* $n-1$ *down to* 0 **do**
  $f \leftarrow f^2$
  $f \leftarrow f \cdot \prod_{k=0}^{e-1} g_k[j](P_k) \cdot h_k[j](Q_k)$, $j \leftarrow j+1$,
  if $s_i \neq 0$ **then**
    $f \leftarrow f \cdot \prod_{k=0}^{e-1} g_k[j](P_k) \cdot h_k[j](Q_k) \cdot \mathcal{S}_{-Q_k}(P_k)$, $j \leftarrow j+1$,
return f.

By direct analogy with Algorithm 5 exchanging P and Q yields another algorithm for positive s.

To estimate the theoretical complexity of Algorithm 5, we need some notations:
– G_{kj} cost of the multiplication by the line function $g_k[j]$,
– $Subs(g_{kj})$ cost of substitution of P_k in the line function $g_k[j]$,
– H_{kj} cost of the multiplication by the line function $h_k[j]$,
– $Subs(h_{kj})$ cost of substitution of Q_k in the line function $h_k[j]$,
– $M_{S_{-Q_i}(P_i)}$ cost of the multiplication by the line $S_{-Q_i}(P_i)$,
– $HW(x)$ the hamming weight of x,
– S_k cost of the squaring in $\mathbb{F}_{p^k}$.
Then, total operations cost of Algorithm 5 is given by

$$\begin{aligned} Cost_{Alg.\ 5} = e \times cost_{S_{-Q_i}(P_i)} + (e-1) \times M_{S_{-Q_i}(P_i)} + \\ \log_2(x) \times (S_k + e \times (G_{kj} + Subs(g_{kj}) + H_{kj} + Subs(h_{kj}))) + \\ HW(x) \times e \times (G_{kj} + Subs(g_{kj}) + H_{kj} + Subs(h_{kj}) + M_{S_{-Q_i}(P_i)}). \quad (4.1) \end{aligned}$$

5. Basic Operations for β-Weil on BLS Curves with Twisted Degree 3

Under this section, basic and special operations for the β-Weil pairing on BLS curves of cubic twist are computed.

Let M, S and I denote the cost of the multiplication, squaring and inversion in $\mathbb{F}_p$, whereas, M_k, S_k, I_k, F_p, E_x denote the cost of the multiplication, squaring, inversion, $p-th$ Frobenius operation and the power of x in F_{p^k} respectively. Let I_{cyc} denote the cost of the inversion in the cyclotomic subgroup G_{ϕ_k}.

5.1. Tower extension

From Table 1, we have a specific value of x as well as the parameter p which defines the field $\mathbb{F}_p$ for each curve BLS-27, BLS-15 and BLS-9. For efficient arithmetic, the extension field $\mathbb{F}_{p^k}$ $(k = 27, 15, 9)$ can be constructed as follows:

$$\begin{aligned} \mathbb{F}_{p^{k/3}} &= \mathbb{F}_{p^i}[\beta]/(\beta^{k/(3i)} - \alpha), \\ \mathbb{F}_{p^k} &= \mathbb{F}_{p^{k/3}}[\omega]/(\omega^3 - \beta), \end{aligned}$$

where α is a primitive root in $\mathbb{F}_{p^i}$ and $i = 1$ for $k = 9, 15$ and $i = 3$ for $k = 27$. We then derived the cited BLS curves $(E) : y^2 = x^3 + b$ from the parameters $p(x), r(x)$ and $t(x)$ for the chosen value x (see table 1). The cubic twisted curve E' of E and their isomorphic mapping ψ_3 are given as follows: $E' : y^2 = x^3 + b\beta^2$ and

$$\begin{aligned} \psi_3 : E'(\mathbb{F}_{p^{k/3}})[r] &\to E(\mathbb{F}_{p^k})[r] \\ (x, y) &\mapsto (\omega^{-2}x, \beta^{-1}y). \end{aligned}$$

5.2. Elliptic curve doubling and elliptic curve addition

Let us consider $T = (\omega^{-2}x_{T'}, \beta^{-1}y_{T'})$, $Q = (\omega^{-2}x_{Q'}, \beta^{-1}y_{Q'})$ given in affine coordinates on the group $E(\mathbb{F}_{p^k})$ such that $T' = (x_{T'}, y_{T'})$ and $Q' = (x_{Q'}, y_{Q'})$ are on the twisted curve E' defined over $\mathbb{F}_{p^{k/3}}$. Let the elliptic curve doubling of T be $2T = (x_{2T}, y_{2T})$ and $P = (x_P, y_P)$ where $x_P, y_P \in \mathbb{F}_p$.

The formulas for computing the doubling and the corresponding line function in affine coordinates are obtained in [13] as follows:

- $\lambda_{T,T} = \frac{3x_T^2}{2y_T} = \frac{3\omega^{-4}x_{T'}^2}{2\beta^{-1}y_{T'}} = \frac{3x_{T'}^2}{2y_{T'}}\omega^{-1} = \lambda_{T',T'}\omega^{-1}$, where $\lambda_{T',T'} = \frac{3x_{T'}^2}{2y_{T'}}$.
- $x_{2T} = \lambda_{T,T}^2 - 2x_T = \lambda_{T',T'}^2\omega^{-2} - 2\omega^{-2}x_{T'} = (\lambda_{T',T'}^2 - 2x_{T'})\omega^{-2} = x_{2T'}\omega^{-2}$, where $x_{2T'} = \lambda_{T',T'}^2 - 2x_{T'}$.
- $y_{2T} = \lambda_{T,T}(x_T - x_{2T}) - y_T = \lambda_{T',T'}\omega^{-1}(x_{T'}\omega^{-2} - x_{2T'}\omega^{-2}) - y_{T'}\beta^{-1} = [\lambda_{T',T'}(x_{T'} - x_{2T'}) - y_{T'}]\beta^{-1} = y_{2T'}\beta^{-1}$ where $y_{2T'} = \lambda_{T',T'}(x_{T'} - x_{2T'}) - y_{T'}$.
- $l_{T,T}(P) = x_P^2 + x_{2T}x_P + x_{2T}^2 - \lambda_{T,T}(y_P - y_{2T}) = x_P^2 + [(x_{2T'}^2 + y_{2T'}\lambda_{T',T'})\beta^{-1} - \lambda_{T',T'}y_P]\omega^{-1} + x_{2T'}x_P\omega^{-2}$.

When considering the affine coordinates $(x, y, t) = (x, y, x^2)$, the previous affine formulas are given by computing the following sequences of operations:
$A = 3t_{T'}$, $B = \frac{1}{2y_{T'}}$, $C = A \cdot B$, $x_{2T'} = C^2 - 2x_{T'}$, $y_{2T'} = C \cdot (x_{T'} - x_{2T'}) - y_{T'}$, $D = x_{2T'}^2$, $E = D\beta^{-1} + C \cdot (y_{2T'}\beta^{-1} - y_P)$, $F = x_{2T'} \cdot x_P$, $x_{2T} = x_{2T'}\omega^{-2}$, $y_{2T} = y_{2T'}\beta^{-1}$, $l_{T,T}(P) = t_P + E\omega^{-1} + F\omega^{-2}$. Therefore the doubling step and line evaluation cost $1I_{k/3} + 3M_{k/3} + 2S_{k/3} + \frac{k}{3}M$. (See [13] for more explanation).

The elliptic curve addition step $(T \neq Q)$ and line evaluation can also be optimized similarly to the above procedure. Let the elliptic curve addition of

T and Q be $R = T + Q = (x_R, y_R)$. The formulas for computing the point addition and the corresponding line function are obtained as follows:

- $\lambda_{T,Q} = \frac{y_Q - y_T}{x_Q - x_T} = \frac{y_{Q'} - y_{T'}}{x_{Q'} - x_{T'}}\beta^{-1}\omega^2 = \frac{y_{Q'} - y_{T'}}{x_{Q'} - x_{T'}}\omega^{-1} = \lambda_{T',Q'}\omega^{-1}$, where $\lambda_{T',Q'} = \frac{y_{Q'} - y_{T'}}{x_{Q'} - x_{T'}}$.
- $x_R = \lambda_{T,Q}^2 - x_Q - x_T = (\lambda_{T',Q'}^2 - x_{Q'} - x_{T'})\omega^{-2} = x_{R'}\omega^{-2}$, where $x_{R'} = \lambda_{T',Q'}^2 - x_{Q'} - x_{T'}$.
- $y_R = \lambda_{T,Q}(x_T - x_R) - y_T = \lambda_{T',Q'}\omega^{-1}(x_{T'}\omega^{-2} - x_{R'}\omega^{-2}) - y_{T'}\beta^{-1} = [\lambda_{T',Q'}(x_{T'} - x_{R'}) - y_{T'}]\beta = y_{R'}\beta^{-1}$ where $y_{R'} = \lambda_{T',Q'}(x_{T'} - x_{Q'}) - y_{T'}$.
- $l_{T,Q}(P) = x_P^2 + x_R x_P + x_R^2 - \lambda_{T,Q}(y_P - y_R) = x_P^2 + [(x_{R'}^2 + y_{R'}\lambda_{T',Q'})\beta^{-1} - \lambda_{T',Q'}y_P]\omega^{-1} + x_{R'}x_P\omega^{-2}$

When considering the affine coordinates $(x, y, t) = (x, y, x^2)$, the previous affine formulas are given by computing sequence of operations:
$A = y_{Q'} - y_{T'}$, $B = \frac{1}{x_{Q'} - x_{T'}}$, $C = A \cdot B$, $x_{R'} = C^2 - x_{Q'} - x_{T'}$, $y_{R'} = C \cdot (x_{T'} - x_{R'}) - y_{T'}$, $D = x_{R'}^2$, $E = D\beta^{-1} + C \cdot (y_{R'}\beta^{-1} - y_P)$, $F = x_{R'} \cdot x_P$, $x_R = x_{R'}\omega^{-2}$, $y_R = y_{R'}\beta^{-1}$, $l_{T,Q}(P) = t_P + E\omega^{-1} + F\omega^{-2}$. Therefore the addition step and line evaluation cost $1I_{k/3} + 3M_{k/3} + 2S_{k/3} + \frac{k}{3}M$.

5.3. Affine sparse multiplication and projective sparse multiplication

The sparse multiplication reduces the cost of the multiplications in $\mathbb{F}_{p^k}$ for Miller's algorithm.

Affine sparse multiplication. In affine coordinates, as we have previously seen, the line function obtained from the elliptic curve doubling and addition steps is

$$l(x, y) = x^2 + (ay + b\beta^{-1})\omega^{-1} + cx\omega^{-2}, \qquad (a, b, c \in \mathbb{F}_{p^{k/3}}),$$

where $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$. Since $\omega^3 = \beta$, then $\omega^{-1} = \beta^{-1}\omega^2$ and $\omega^{-2} = \beta^{-1}\omega$ hence,

$$l(x, y) = A + B\omega + C\omega^2, \qquad (5.1)$$

where $A = x^2, B = cx\beta^{-1}$ and $C = ay\beta^{-1} + b\beta^{-2}$.

The sparse multiplication consists of computing $f \cdot l(P)$ with f in $\mathbb{F}_{p^k}$ and it is evaluated through Algorithm 6.

Algorithm 6: Affine Sparse Multiplication (ASM) in $\mathbb{F}_{p^k}$

Input: $a = a_0 + a_1\omega + a_2\omega^2$, where $a_0 \in \mathbb{F}_p$ and $a_1, a_2 \in \mathbb{F}_{p^{k/3}}$
$b = b_0 + b_1\omega + b_2\omega^2$ where $b_0, b_1, b_2 \in \mathbb{F}_{p^{k/3}}$
Output: $a \cdot b \in \mathbb{F}_{p^k}$

1 $A \leftarrow a_0b_0$, $B \leftarrow a_1b_1$, $C \leftarrow a_2b_2$,
2 $D_0 \leftarrow a_0 + a_1$, $D_1 \leftarrow a_1 + a_2$, $D_2 \leftarrow a_0 + a_1 + a_2$,
3 $E_0 \leftarrow b_0 + b_1$, $E_1 \leftarrow b_1 + b_2$, $E_2 \leftarrow b_0 + b_1 + b_2$,
4 $F_0 \leftarrow D_0E_0 - (A + B)$, $F_1 \leftarrow D_1E_1 - (B + C)$,
$F_2 \leftarrow D_2E_2 - (A + C + F_0 + F_1)$,
5 $G_0 \leftarrow A + F_1\beta$, $G_1 \leftarrow F_0 + C\beta$,
6 return $c \leftarrow G_0 + G_1\omega + F_2\omega^2$.

The Affine Sparse Multiplication cost $5M_{k/3} + \frac{k}{3}M$ in the base field (instead of $6M_{k/3}$ multiplications with the Karatsuba formulas).

Moreover, the vertical line function from elliptic curve is given as follows

$$\mathcal{V}(x, y) = x - a,$$

where a belongs to $\mathbb{F}_p$. We substitute Q in the previous equation and obtain $\mathcal{V}(Q) = \beta^{-1}x_{Q'}\omega - a$. The sparse multiplication $f \cdot \mathcal{V}(Q)$ for f in $\mathbb{F}_{p^k}$ can be calculated by Algorithm 7.

Algorithm 7: Sparse Multiplication with vertical line in $\mathbb{F}_{p^k}$

Input: $a = a_0 + a_1\omega$, where $a_0 \in \mathbb{F}_p$ and $a_1 \in \mathbb{F}_{p^{k/3}}$
$b = b_0 + b_1\omega + b_2\omega^2$ where $b_0, b_1, b_2 \in \mathbb{F}_{p^{k/3}}$
Output: $a \cdot b \in \mathbb{F}_{p^k}$

$A \leftarrow a_0b_0$, $B \leftarrow a_1b_1$, $C \leftarrow a_0b_2$, $D \leftarrow a_1b_2$,
$E \leftarrow a_0 + a_1$, $F \leftarrow b_0 + b_1$, $H \leftarrow EF - (A + B)$,
$I \leftarrow A + D\beta$, $J \leftarrow B + C$,
return $c \leftarrow I + H\omega + J\omega^2$.

The affine Sparse Multiplication cost $3M_{k/3} + 2(\frac{k}{3}M)$ in the base field (instead of $6M_{k/3}$ multiplications with the Karatsuba formulas).

Projective sparse multiplication. In projective coordinates, the line function from elliptic curve doubling and addition steps is given in [32] as follows

$$l(x, y) = ax^2 + bx + cy + d,$$

where a, b, c and d are in $\mathbb{F}_p$. The point $Q = (x, y)$ in $E(\mathbb{F}_{p^k})$ can be viewed as $\psi_3(Q') = (\omega^{-2}x_{Q'}, \beta^{-1}y_{Q'})$ for Q' in $E(\mathbb{F}_{p^{k/3}})$. We substitute Q in the previous equation and obtain $l(Q) = A + B\omega + C\omega^2$, where $A = cy_{Q'}\beta^{-1} + d, B = bx_{Q'}\beta^{-1}$ and $C = ax_{Q'}^2\beta^{-2}$ are in $\mathbb{F}_{p^{k/3}}$. We did not find an efficient computation method for the following multiplication $f \cdot l(Q)$ where $f \in \mathbb{F}_{p^k}$.

The inverse of the vertical line. From Eq. 3.4, the inverse of the vertical line $1/\mathcal{V}_Q(P)$ becomes $\mathcal{S}_Q(P) = x_P^2 + x_Qx_P + x_Q^2$. For $Q = (\omega^{-2}x_{Q'}, \beta^{-1}y_{Q'})$, with $(x_{Q'}, y_{Q'})$ in $E(\mathbb{F}_{p^{k/3}})$ we have, $\mathcal{S}_Q(P) = x_P^2 + x_{Q'}x_P\omega^{-2} + x_{Q'}^2(\beta\omega)^{-1}$. This cost $1S_{k/3} + k/3M$.

p^i-th Frobenius map for the line function. From Eq. 5.1 the line functions for computing $f_{x,Q}$ are represented as :

$$l(x, y) = A + B\omega + C\omega^2,$$

where, $A \in \mathbb{F}_p$ and $B, C \in \mathbb{F}_{p^{k/3}}$. The p^i-th Frobenius map is $l^{p^i}(x, y) = A + B^{p^i}\omega^{p^i} + C\omega^{2p^i}$, since $A \in \mathbb{F}_p$. The cost of Frobenius map for the extension field element is at most the degree of field extension, so the cost of p^i-th Frobenius map for B and C are $2\frac{k}{3}M$, since $B, C \in \mathbb{F}_{p^{k/3}}$. Then for a given ω^{p^i} and ω^{2p^i}, the p^i-th Frobenius map for line function cost $2\frac{k}{3}M$.

p^i-th Frobenius map for a point Q..

$$\pi_{p^i}(Q) = \pi_{p^i}(\omega^{-2}x_{Q'}; \beta^{-1}y_{Q'}) = \left((\omega^{p^i})^{-2}x_{Q'}^{p^i}; (\beta^{p^i})^{-1}y_{Q'}^{p^i}\right)$$

For the given ω^{-p^i} and β^{-p^i}, the computational costs of $\pi_{p^i}(Q)$ is the total cost of the p^i-th Frobenius map for $x_{Q'}$ and $y_{Q'}$.

6. Application on BLS-27: theoretical costs of β-Weil computation

We used the ideas of storage and multifunction techniques to evaluate the β-Weil pairing on BLS-27 either in serial or parallel computation.

Proposition 6.1. *For pairing-friendly curves of embedding degrees $k = 27, 15$ and 9, with the extended Miller function $h(z) = x - z$ such that $h(p) = 0 \mod r$, the relation $[p]P = [x]P$ holds, for any P in $\mathbb{G}_1$ or $\mathbb{G}_2$ of order r.*

Proof. : By hypothesis, $h(p) = 0 \mod r$, then $x = p \mod r$, and $[x]P = [p]P$, since P is of order r. □

The proposed β-Weil pairing formula for BLS-27 in Table 1 is

$$\beta_{27}(P,Q) = \Big[\prod_{i=0}^{8}(f_{x,Q_i}(P_i)\cdot f_{-x,P_i}(Q_i))\cdot\prod_{i=0}^{8}\mathcal{V}_{P_{i+1}}(Q_i)\Big]^{p^9-1}.$$

The basic operations used to evaluate the theoretical cost of the β-Weil pairing on BLS-27 are recapitulated in the following Table 2.

TABLE 2. Costs of arithmetic operations in a tower extension field $\mathbb{F}_{p^{27}}$ and special operations in the β-Weil pairing computations on BLS-27 in this work and [17].

Field	Multiplication	Squaring	Inversion
$\mathbb{F}_{p^3}$	$M_3 = 6M$	$S_3 = 5M$	$I_3 = I + 9M + 2S$
$\mathbb{F}_{p^9}$	$M_9 = 36M$	$S_9 = 25M$	$I_9 = I + 63M + 12S$
$\mathbb{F}_{p^{27}}$	$M_{27} = 216M$	$S_{27} = 125M$	$I_{27} = I + 387M + 62S$
$\mathbb{F}_{p^{27}}$ **Arithmetic**		**Operation count** $(S = M)$	
Affine sparse multiplication		$189M$	
Projective sparse multiplication		$216M$	
p^i Frobenius ([17] table 2)		$18M$ for $i = 3, 6, 9$ and $26M$ otherwise i	
$E'(\mathbb{F}_{p^9})$ **Arithmetic**		**Operation count** $(S = M)$	
Point doubling with line		$I_9 + 2S_9 + 3M_9 = 233M + I$	
Point addition with line		$I_9 + 2S_9 + 3M_9 = 233M + I$	
Substitution of P in the line $l(x,y) = x^2 + (a\beta + by)\omega + cx\omega^2$		$9M$	
$\pi_{p^i}(Q)$ ([17] table 2)		$12M$ for $i = 3, 6, 9$ and $16M$ otherwise	
$E(\mathbb{F}_p)$ **Arithmetic** (see [32], section 4)		**Operation count** $(S = M)$	
Point doubling with line		$6M + 7S = 13M$	
Mixed point addition with line		$13M + 3S = 16M$	
Substitution Q for line $l(x,y) = (ax + b)x + cy + d$		$27M$	

6.1. Serial computation of the β-Weil pairing over BLS-27

In the first paragraph, we evaluate the theoretical cost of the line functions for each of the Miller functions. In the second paragraph, we estimate the cost of $A = \prod_{i=0}^{8}(f_{x,Q_i}(P_i) \cdot f_{-x,P_i}(Q_i))$ using Algorithm 5 and the line functions previously obtained. After we evaluate the product of the vertical lines and end with the final powering $p^9 - 1$.

On the first hand, we compute in projective coordinates and store the line functions for each Miller-Lite function f_{-x,P_i} where, $x = -2^{51} - 2^{31} - 2^{21} - 2^8 - 2^4$, and $P_i = [p^i]P$. This is done through Algorithm 3: $CSL([p^i]P, -x) = h_i$. Each Miller's function is composed by 51 doubling steps $(13M)$ and 4 mixed addition steps $(16M)$. So for each $0 \leq i \leq 8$, the addition and doubling steps cost $51 \cdot 13M + 4 \cdot 16M = 727M$. Since $[p^i]P$ is equal to $[-xp^{i-1}]P$ (see Proposition 6.1) and that we obtain $[-xp^{i-1}]P$ after computing $CSL([p^{i-1}]P, -x)$, we do not need to compute again $[p^i]P$. For mixed addition coordinate (see [32]), it is required to convert $[p^i]P = (X_i : Y_i : Z_i)$ from projective to affine coordinates $(X_i \cdot Z_i^{-1}, Y_i \cdot Z_i^{-1})$ for a cost of $I + 2M$ for each i $(1 \leq i \leq 8)$. Note that for $i = 0$, $P_0 = P$ is given in each coordinate. Hence, the total cost of all line functions computation for all Miller-Lite functions is $9 \cdot 727M + 8(I + 2M) = 6559M + 8I$.
On the other hand, we compute in affine coordinates and store the line functions for computing $f_{x,Q}$. This is done through Algorithm 3: $CSL(Q, x) = g_0$. The Full Miller's function is composed by 51 doubling steps $(233M + I)$ and 4 mixed addition steps $(233M + I)$. This yields $51(233M + I) + 4(233M + I) = 12815M + 55I$. From x is $-2^{51} - 2^{31} - 2^{21} - 2^8 - 2^4$, $f_{x,Q}$ is made of 55 line functions. Since the p^i-th Frobenius map for line function is $18M$, the cost of one $CSLFrob(g_0, i)$ execution for $f_{x,\pi_{p^{8-i}}(Q)}$ is $55 \cdot 18M = 990M$. Therefore, the total cost of all line functions computation for all Full-Miller's functions is $(12815M + 55I) + 8 \cdot 990M = 20735M + 55I$.
Hence, the total cost of all line functions computation for $A = \prod_{i=0}^{8}(f_{x,Q_i}(P_i) \cdot f_{-x,P_i}(Q_i))$ is $(6559M + 8I) + (20735M + 55I) =$ **27294 M+ 63 I** .

Secondly, we evaluate all line functions at a given point and compute the product of Miller's functions by using Algorithm 5. From Table 2, the computational costs of $\pi_{p^i}(Q)$ for $1 \leq i \leq 8$ is $2 \cdot 12M + 6 \cdot 16M = 120M$. Since each $\mathcal{S}_{-Q_i}(P_i)$ (Eq. 3.4) costs $S_9 + 9M = 34M$ and the multiplication by $\mathcal{S}_{-Q_i}(P_i)$ in $\mathbb{F}_{p^{27}}$ costs $189M$. From Eq. 4.1, the execution cost for A is $9 \cdot 34M + 8 \cdot 189M + 51 \cdot 125 + 51 \cdot 9 \cdot (189M + 9M + 216M + 27M) + 4 \cdot 9 \cdot (189M + 9M + 216M + 27M + 189M) = 233292M$. Therefore the total cost is $120M + 233292M =$ **233412 M**.

Thirdly, we compute $B = A \cdot \prod_{i=0}^{8} \mathcal{V}_{P_{i+1}}(Q_i)$. To have B, we multiply A by each $\mathcal{V}_{P_{i+1}}(Q_i)$ for $0 \leq i \leq 8$ for a cost of $9 \cdot 126M =$ **1134 M** (see Algorithm 7). There is no need to evaluate the point $P_9 = [p^9]Q$ as we obtain P_9 after computing f_{-x,P_8}.

Finally, $(B)^{p^9-1}$ is made of one inversion, one multiplication and one p^9-Frobenius in $\mathbb{F}_{p^{27}}$, at the cost of $I_{27} + M_{27} + 18M =$ **683 M+I**.

The total cost of serial computation of the β-Weil pairing is $(27294M + 63I) + (233412M) + (1134M) + (683M + I) =$ **262523 M+64 I**.

6.2. Parallel computation of $\beta_{27}(P, Q)$ using 3 processors

Here we parallelise the evaluation of $\beta_{27}(P, Q)$ and find its theoretical cost. In the first step, each processor computes and stores 6 line functions. For step 2, each processor evaluates the stored line functions using the multifuction technique. Whereas step 3 computes the additional factors of $\beta_{27}(P, Q)$ and the last step combines each result. Since $P_i = [p^i]P$ and $Q_i = \pi_{p^{8-i}}(Q)$ for $0 \leq i \leq 8$,

$$\begin{aligned}
\beta_{27}(P,Q) \;=\; & [f_{x,\pi_{p^8}(Q)}(P) \cdot f_{-x,P}(\pi_{p^8}(Q)) \cdot f_{x,\pi_{p^5}(Q)}(P_3) \cdot f_{-x,P_3}(\pi_{p^5}(Q)) \cdot f_{x,\pi_{p^2}(Q)}(P_6) \\
& \cdot f_{-x,P_6}(\pi_{p^2}(Q)) \cdot \mathcal{V}_{P_1}(\pi_{p^8}(Q)) \cdot \mathcal{V}_{P_4}(\pi_{p^5}(Q)) \cdot \mathcal{V}_{P_7}(\pi_{p^2}(Q)) \\
& \cdot f_{x,\pi_{p^7}(Q)}(P_1) \cdot f_{-x,P_1}(\pi_{p^7}(Q)) \cdot f_{x,\pi_{p^4}(Q)}(P_4) \cdot f_{-x,P_4}(\pi_{p^4}(Q)) \cdot f_{x,\pi_p(Q)}(P_7) \\
& \cdot f_{-x,P_7}(\pi_p(Q)) \cdot \mathcal{V}_{P_2}(\pi_{p^7}(Q)) \cdot \mathcal{V}_{P_5}(\pi_{p^4}(Q)) \cdot \mathcal{V}_{P_8}(\pi_p(Q)) \\
& \cdot f_{x,\pi_{p^6}(Q)}(P_2) \cdot f_{-x,P_2}(\pi_{p^6}(Q)) \cdot f_{x,\pi_{p^3}(Q)}(P_5) \cdot f_{-x,P_5}(\pi_{p^3}(Q)) \cdot f_{x,Q}(P_8) \\
& \cdot f_{-x,P_8}(Q) \cdot \mathcal{V}_{P_3}(\pi_{p^6}(Q)) \cdot \mathcal{V}_{P_6}(\pi_{p^3}(Q)) \cdot \mathcal{V}_{P_9}(Q)]^{p^9-1}.
\end{aligned}$$

That is,

$$\begin{aligned}
\beta_{27}(P,Q) \;=\; & [\{f_{x,\pi_{p^6}(Q)}(P) \cdot f_{-x,P}(\pi_{p^6}(Q)) \cdot f_{x,\pi_{p^3}(Q)}(P_3) \cdot f_{-x,P_3}(\pi_{p^3}(Q)) \cdot f_{x,Q}(P_6) \\
& \cdot f_{-x,P_6}(Q) \cdot \mathcal{V}_{P_1}(\pi_{p^6}(Q)) \cdot \mathcal{V}_{P_4}(\pi_{p^3}(Q)) \cdot \mathcal{V}_{P_7}(Q)\}^{p^2} \\
& \cdot \{f_{x,\pi_{p^6}(Q)}(P_1) \cdot f_{-x,P_1}(\pi_{p^6}(Q)) \cdot f_{x,\pi_{p^3}(Q)}(P_4) \cdot f_{-x,P_4}(\pi_{p^3}(Q)) \cdot f_{x,Q}(P_7) \\
& \cdot f_{-x,P_7}(Q) \cdot \mathcal{V}_{P_2}(\pi_{p^6}(Q)) \cdot \mathcal{V}_{P_5}(\pi_{p^3}(Q)) \cdot \mathcal{V}_{P_8}(Q)\}^{p} \\
& \cdot f_{x,\pi_{p^6}(Q)}(P_2) \cdot f_{-x,P_2}(\pi_{p^6}(Q)) \cdot f_{x,\pi_{p^3}(Q)}(P_5) \cdot f_{-x,P_5}(\pi_{p^3}(Q)) \cdot f_{x,Q}(P_8) \\
& \cdot f_{-x,P_8}(Q) \cdot \mathcal{V}_{P_3}(\pi_{p^6}(Q)) \cdot \mathcal{V}_{P_6}(\pi_{p^3}(Q)) \cdot \mathcal{V}_{P_9}(Q)]^{p^9-1}.
\end{aligned}$$

For parallel computation using 3 processors, $\beta_{27}(P, Q)$ can be regarded as

$$\beta_{27}(P,Q) = (X^{p^2} \cdot Y^p \cdot Z)^{p^9-1},$$

where

$$\begin{aligned}
X &= f_{x,\pi_{p^6}(Q)}(P) \cdot f_{-x,P}(\pi_{p^6}(Q)) \cdot f_{x,\pi_{p^3}(Q)}(P_3) \cdot f_{-x,P_3}(\pi_{p^3}(Q)) \cdot f_{x,Q}(P_6) \cdot f_{-x,P_6}(Q) \cdot H_1, \\
Y &= f_{x,\pi_{p^6}(Q)}(P_1) \cdot f_{-x,P_1}(\pi_{p^6}(Q)) \cdot f_{x,\pi_{p^3}(Q)}(P_4) \cdot f_{-x,P_4}(\pi_{p^3}(Q)) \cdot f_{x,Q}(P_7) \cdot f_{-x,P_7}(Q) \cdot H_2, \\
Z &= f_{x,\pi_{p^6}(Q)}(P_2) \cdot f_{-x,P_2}(\pi_{p^6}(Q)) \cdot f_{x,\pi_{p^3}(Q)}(P_5) \cdot f_{-x,P_5}(\pi_{p^3}(Q)) \cdot f_{x,Q}(P_8) \cdot f_{-x,P_8}(Q) \cdot H_3
\end{aligned}$$

and

$$\begin{aligned}
H_1 &= \mathcal{V}_{P_1}(\pi_{p^6}(Q)) \cdot \mathcal{V}_{P_4}(\pi_{p^3}(Q)) \cdot \mathcal{V}_{P_7}(Q), \\
H_2 &= \mathcal{V}_{P_2}(\pi_{p^6}(Q)) \cdot \mathcal{V}_{P_5}(\pi_{p^3}(Q)) \cdot \mathcal{V}_{P_8}(Q), \\
H_3 &= \mathcal{V}_{P_3}(\pi_{p^6}(Q)) \cdot \mathcal{V}_{P_6}(\pi_{p^3}(Q)) \cdot \mathcal{V}_{P_9}(Q).
\end{aligned}$$

Step 1.

1. The 1^{st} processor computes and stores $CSL([p^i]P, -x) = h_i$ for $0 \leq i \leq 8$ for a cost of $9 \cdot 727M + 8(I + 2M) = 6559M + 8I$ and $CSL(Q, x) = g_0$ for a cost of $12815M + 55I$. The total cost is **19374 M+63 I**.
2. The 2^{nd} processor computes and stores $\pi_{p^3}(Q)$ and $CSL(\pi_{p^3}(Q), x) = g_3$ for a cost of $12M + 12815M + 55I = 12827M + 55I$.

3. The 3^{rd} processor computes and stores $\pi_{p^6}(Q)$ and $CSL(\pi_{p^6}(Q), x) = g_6$ for the same cost as the 2^{nd} processor.

Step 2.

1. From Eq. 4.1, the 1^{st} processor computes $X_1 = EPM\big(\big[(g_6, P_0), (g_3, P_3), (g_0, P_6), (h_0, \pi_{p^6}(Q)), (h_3, \pi_{p^3}(Q)), (h_6, Q)\big], x\big)$ for a cost of $3 \cdot 34M + 2 \cdot 189M + 51S_{27} + 51 \cdot 3 \cdot (189M + 9M + 216M + 27M) + 4 \cdot 3 \cdot (189M + 9M + 216M + 27M + 189M) =$ **81922 M**.
2. The 2^{nd} processor computes $Y_2 = EPM\big(\big[(g_6, P_1), (g_3, P_4), (g_0, P_7), (h_1, \pi_{p^6}(Q)), (h_4, \pi_{p^3}(Q)), (h_7, Q)\big], x\big)$ for the same cost as the 1^{st} processor.
3. The 3^{rd} processor computes $Z_3 = EPM\big(\big[(g_6, P_2), (g_3, P_5), (g_0, P_8), (h_2, \pi_{p^6}(Q)), (h_5, \pi_{p^3}(Q)), (h_8, Q)\big], x\big)$ for the same cost as the 1^{st} processor.

Step 3.

1. The 1^{st} processor computes $X = X_1 \cdot H_1 = X_1 \cdot \mathcal{V}_{P_1}(\pi_{p^6}(Q)) \cdot \mathcal{V}_{P_4}(\pi_{p^3}(Q)) \cdot \mathcal{V}_{P_7}(Q)$ at cost of $3 \cdot 126M =$ **378 M**.
2. The 2^{nd} processor computes Y for the same cost as the 1^{st} processor.
3. The 3^{rd} processor computes Z for the same cost as the 1^{st} processor.

Step 4.

Since $\beta_{27}(P, Q) = ((X^p \cdot Y)^p \cdot Z)^{p^9-1}$, in the final step one processor computes two $p-$ and one p^9-Frobenius maps, three multiplications and one inversion in $\mathbb{F}_{p^{27}}$ which yields $2 \cdot 26M + 18M + 3M_{27} + I_{27} =$ **1167 M+I**.

The total cost of parallel computation of the β-Weil pairing using 3 processors is $(19374M + 63I) + 81922M + 378M + (1167M + I) =$ **102841 M+64I**.

The other cases of the β-Weil pairing on the BLS-15 and BLS-9 curves are computed in the similar ways the complete version will be published in the in Mathematics and Computer Science (MCS).

7. Comparison

In this section, we compare the theoretical costs of the optimal Ate pairing (see Appendix B, C, D), the original β-Weil pairing (see Appendix A) and the proposed β-Weil pairing (Corollary 3.4) on the BLS-27, BLS-15, BLS-9 curves. The parallel computation of the optimal Ate pairing is obtained when parallelising the computation of Miller loop only [8], since to date there is not a way to parallelise the final exponentiation. Given that in the literature there is no parallel computation of the optimal Ate pairing on the aforementioned curves, we estimate the costs by dividing a Miller loop cost by the number of processors (see B.1) and add this to the final exponentiation cost. Then our theoretical results reduce the number of multiplication operations on the prime field in β-Weil pairing for BLS family with $k = 27.15$ and 9 about 44.78%, 49.07% and 38.49%, as well as the number of divisions of about 87.1% 78.8% and 61.2% respectively for serial computation. The optimal Ate pairing when compared with the new β-Weil pairing in the sequential case, it is approx. 33% more efficient for $k = 27$ and approx. 17% more efficient for $k = 9$. Also, the two pairing types for $k = 15$ have pretty much the same

execution time. However, the proposed β-Weil pairing on curve with $k = 15$ is competitive to optimal Ate pairing. Indeed, since the final exponentiation in the optimal Ate pairing is expensive and it is about $5, 4$ and 2 times the Full Miller functions for $k = 27, 15$ and 9 respectively with one extra Full Miller function the optimal Ate pairing is about $6, 5$ and 3 times the Full Miller functions for $k = 27, 15$ and 9 respectively. In the other hand without the Miller Lite functions and the simple final exponentiation, the new β-Weil pairing is made of $9, 5$ and 3 the Full Miller functions that need to be evaluate by only one processor for $k = 27, 15$ and 9 respectively. This explain the difference between the two pairings on the BLS-27, BLS-15 and BLS-9.

In the parallel computation with 3 processors, the β-Weil pairing on BLS-27, BLS-15 and BLS-9 is faster than the optimal Ate pairing. See the Table 3 for a complete comparison.

TABLE 3. Theoretical cost of the optimal Ate pairing, the original β-Weil pairing (without storage technique and multifunction technique) and the proposed β-Weil pairing.

curve	pairing	Serial computation	Parallel computation (with 3 processors)
BLS-27 at 256-bit	Optimal Ate [17]	$176,881M + 56I$	$156,301M + 20I$
	original β-Weil pairing [10]	$475,463M + 497I$	$163,815M + 167I$
	Proposed β-Weil pairing	**262,523 M + 64 I**	**102,841 M+64 I**
BLS-15 at 192-bit	Optimal Ate [17]	$91,469M + 81I$	$81,220M + 41I$
	original β-Weil pairing [10]	$177,657M + 401I$	-
	Proposed β-Weil pairing	**90,477 M + 85 I**	**46,666 M+ 85 I**
BLS-9 at 128-bit	Optimal Ate [17]	$22,365M + 78I$	$18,379M + 40I$
	original β-Weil pairing [10]	$43,873M + 232I$	-
	Proposed β-Weil pairing	**26,984 M+90 I**	**11,414 M+78 I**

Table 4 gives the number of $\mathbb{F}_p$ elements required to store for the β-Weil pairing computation. The total storage of elements is determined by the following formula $a_1b_1c_1 + a_2b_2c_2$, where,
$-$ a_1 and a_2 are the number of stored extended Miller functions f_{p,h,P_i} and f_{p,h,Q_i} respectively,
$-$ b_1 and b_2 are the number of stored lines in f_{p,h,P_i} and f_{p,h,Q_i} respectively,
$-$ c_1 and c_2 are the number of $\mathbb{F}_p$ elements of line coefficients in f_{p,h,P_i} and f_{p,h,Q_i} respectively.

TABLE 4. Number of $\mathbb{F}_p$ elements required to be stored for our method.

Curve	Serial computation	Parallel computation (with 3 processors)
BLS-27	$9 \cdot 55 \cdot 4 + 9 \cdot 55 \cdot 27 = 15,345$	$9 \cdot 55 \cdot 4 + 3 \cdot 55 \cdot 27 = 6,435$
BLS-15	$5 \cdot 80 \cdot 4 + 5 \cdot 80 \cdot 15 = 7,600$	$5 \cdot 80 \cdot 4 + 3 \cdot 80 \cdot 15 = 5,200$
BLS-9	$3 \cdot 77 \cdot 4 + 3 \cdot 77 \cdot 9 = 3,003$	$3 \cdot 77 \cdot 4 + 2 \cdot 77 \cdot 9 = 2,310$

8. Conclusion

In this paper, we extended the work of Kinoshita and Suzuki [1] by providing a new formula for the β-Weil pairing on curves with odd embedding degrees. This formula involves vertical line functions useful for the sparse multiplications during Miller's loop. For faster computation of the proposed β-Weil pairing we compute and store line functions for some Miller's functions that are reused to find other line functions for other Miller's functions. The multifunction technique evaluates the product of n Miller's functions and only requires a single squaring in the extension field per iteration instead of n squarings in the naive way. Then our theoretical results give faster β-Weil computation on the BLS family with $k = 9, 15$ and 27 than the original β-Weil pairing on the same curves. The proposed β-Weil pairing on curve with $k = 15$ is competitive to optimal Ate pairing. The implementation results of the proposed methods and the original β-Weil pairings on BLS-9, BLS-15 and BLS-27 curves are left for future work to confirm the theoretical results.

References

[1] Kinoshita K and Suzuki K. Accelerating Beta Weil pairing with precomputation and multi-pairing techniques. In Kazumaro Aoki and Akira Kanaoka, editors, *Advances in Information and Computer Security - 15th International Workshop on Security, IWSEC 2020, Fukui, Japan, September 2-4, 2020, Proceedings*, volume 12231 of *Lecture Notes in Computer Science*, pages 261–281. Springer, 2020.

[2] Silvermann J.H. *The Arithmetic of elliptic curves*, volume 106 of graduate texts in Mathematics. Springer-Verlag, 1986.

[3] Boneh D and Matthew KF. Identity-based encryption from the Weil pairing. In Kilian Joe, editor, *In Advances in cryptology Crypto2001*, volume 2139, pages 213–229. Springer Berlin Heidelberg, 2001.

[4] Boneh D, Lynn B, and Shacham H. Short signatures from the Weil pairing. *J Cryptology*, 17(4):297–319, 2004.

[5] Joux A. A one round protocol for tripartite diffie-hellman. In *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings*, volume 1838, pages 385–394, 2000.

[6] Scott M and Barreto PSLM. Compressed pairings. In Franklin Matt, editor, *In Advances in cryptology Crypto2004*, volume 3152, pages 140–156. Springer-Verlag, 2004.

[7] Vercauteren F. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461, 2010.

[8] Aranha DF, Knapp E, Menezes A, and Rodríguez-Henríquez F. Parallelizing the Weil and Tate pairings. In Liqun Chen, editor, *Cryptography and Coding - 13th IMA International Conference, IMACC 2011, Oxford, UK, December 12-15, 2011. Proceedings*, volume 7089 of *Lecture Notes in Computer Science*, pages 275–295. Springer, 2011.

[9] Aranha DF, Castañeda LF, Knapp E, Menezes A, and Rodríguez-Henríquez F. Implementing pairings at the 192-bit security level. In Michel Abdalla and Tanja Lange, editors, *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, volume 7708 of *Lecture Notes in Computer Science*, pages 177–195. Springer, 2012.

[10] Fouotsa E., Pecha A., and El Mrabet N. Beta Weil pairing revisited. *Afrika Matematika.*, 30:371–388, 2019.

[11] Zhao C, Xie D, Zhang F, Zhang J, and Chen BL. Computing bilinear pairings on elliptic curves with automorphisms. *Des. Codes Cryptogr.*, 58(1):35–44, 2011.

[12] Galbraith SD, Lin X, and Scott M. Endomorphisms for faster elliptic curve cryptography on a large class of curves. *J Cryptology*, 24(3):446–469, 2011.

[13] Zhang X and Lin D. Analysis of optimum pairing products at high security levels. In Steven D. Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, volume 7668 of *Lecture Notes in Computer Science*, pages 412–430. Springer, 2012.

[14] Barbulescu R, El Mrabet N, and Ghammam L. A taxonomy of pairings, their security, their complexity. *IACR Cryptology ePrint Arch.*, 2019:485, 2019.

[15] Miller SV. The Weil pairing, and its efficient calculation. *J Cryptology*, 17(4):235–261, 2004.

[16] Feng QY, Ming TC, Baoan G, and Zhi XM. Super-optimal pairings. In *Mechanical Engineering, Materials and Energy II*, volume 281 of *Applied Mechanics and Materials*, pages 127–133. Trans Tech Publications Ltd, 3 2013.

[17] Fouotsa E, El Mrabet N, and Pecha A. Optimal Ate pairing on elliptic curves with embedding degree 9, 15 and 27. *journal of Groups, Complexity, Cryptology*, 12, 2020.

[18] Clarisse R, Duquesne S, and Sanders O. Curves with fast computations in the first pairing group. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *Cryptology and Network Security - 19th International Conference, CANS 2020, Vienna, Austria, December 14-16, 2020, Proceedings*, volume 12579 of *Lecture Notes in Computer Science*, pages 280–298. Springer, 2020.

[19] Lavice A, El Mrabet N, Berzati A, Rigaud JB, and Proy J. Hardware implementations of pairings at updated security levels. In Vincent Grosso and Thomas Pöppelmann, editors, *Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers*, volume 13173 of *Lecture Notes in Computer Science*, pages 189–209. Springer, 2021.

[20] El Mrabet N and Fouotsa E. Failure of the point blinding countermeasure against fault attack in pairing-based cryptography. In Said El Hajji, Abderrahmane Nitaj, Claude Carlet, and El Mamoun Souidi, editors, *Codes, Cryptology, and Information Security - First International Conference, C2SI 2015, Rabat, Morocco, May 26-28, 2015, Proceedings - In Honor of Thierry Berger*, volume 9084 of *Lecture Notes in Computer Science*, pages 259–273. Springer, 2015.

[21] Blömer J, Gomes da Silva R, Günther P, Krämer J, and Seifert JP. A practical second-order fault attack against a real-world pairing implementation. *IACR Cryptol. ePrint Arch.*, page 543, 2014.

[22] Weng J, Dou Y, Chuangui Ma, and El Mrabet N. Fault attacks against the miller algorithm in hessian coordinates. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, volume 7537 of *Lecture Notes in Computer Science*, pages 102–112. Springer, 2011.

[23] Scott M. Pairing implementation revisited. *IACR Cryptology ePrint Arch.*, page 77, 2019.

[24] Scott M. Computing the Tate pairing. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, pages 293–304, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[25] Azebaze GL, Fouotsa E, El Mrabet N, and Pecha A. Sage code for the verification of various algorithms/ formulas and bilinearity of pairings. In *http://www.emmanuelfouotsa-prmais.org/Portals/22/codeBetaWeil.zip*, 2021.

[26] Guillevic A. A short-list of pairing-friendly curves resistant to special TNFS at the 128-bit security level. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 535–564. Springer, 2020.

[27] Scott M and Guillevic A. A new family of pairing-friendly elliptic curves. In Lilya Budaghyan and Francisco Rodríguez-Henríquez, editors, *Arithmetic of Finite Fields - 7th International Workshop, WAIFI 2018, Bergen, Norway, June 14-16, 2018, Revised Selected Papers*, volume 11321 of *Lecture Notes in Computer Science*, pages 43–57. Springer, 2018.

[28] Barbulescu R and Duquesne S. Updating key size estimations for pairings. *J Cryptology*, 32(4):1298–1336, 2019.

[29] Barreto PSLM, Lynn B, and Scott M. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, volume 2576 of *Lecture Notes in Computer Science*, pages 257–267. Springer, 2002.

[30] Enge A and Milan J. Implementing cryptographic pairings at standard security levels. In Rajat Subhra Chakraborty, Vashek Matyas, and Patrick Schaumont, editors, *Security, Privacy, and Applied Cryptography Engineering - 4th International Conference, SPACE 2014, Pune, India, October 18-22, 2014. Proceedings*, volume 8804 of *Lecture Notes in Computer Science*, pages 28–46. Springer, 2014.

[31] Lin X, Zhao C, Zhang F, and Wang Y. Computing the Ate pairing on elliptic curves with embedding degree $k = 9$. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 2007(9):2387–2393, 2008.

[32] Costello C, Lange T, and Naehrig M. Faster pairing computations on curves with high-degree twists. In *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, pages 224–242, 2010.

[33] Aranha DF, Knapp E, Menezes A, and Rodríguez-Henríquez F. Parallelizing the Weil and Tate pairings. In Liqun Chen, editor, *Cryptography and Coding - 13th IMA International Conference, IMACC 2011, Oxford, UK, December 12-15, 2011. Proceedings*, volume 7089 of *Lecture Notes in Computer Science*, pages 275–295. Springer, 2011.

[34] Hayashida D, Hayasaka K, and Teruya T. Efficient final exponentiation via cyclotomic structure for pairings over families of elliptic curves. *J Cryptology ePrint Arch.*, 2020:875, 2020.

Appendix A. Original β-Weil pairing on BLS curves with embedding degree $27, 15$ and 9.

In this section, we find the theoretical cost of the original β-Weil pairing computation on BLS-27, BLS-15, and BLS-9 curves without storage technique and multifunction technique.

The original β-Weil pairing (see theorem 2.1) on BLS-27 for seed $x = -2^{51} - 2^{31} - 2^{21} - 2^8 - 2^4$ is given as:

$$\begin{aligned}
\beta_{27}(P,Q) &= \prod_{i=0}^{8} \left(\frac{f_{x,Q}([p^i]P)}{f_{x,[p^i]P}(Q)} \right)^{(p^{8-i})(p^9-1)} \\
&= \Big[\big(f_{x,Q}^{p^8}(P) \cdot f_{x,Q}^{p^7}([p]P) \cdot f_{x,Q}^{p^6}([p^2]P) \cdot f_{x,Q}^{p^5}([p^3]P) \cdot f_{x,Q}^{p^4}([p^4]P) \\
&\quad \cdot f_{x,Q}^{p^3}([p^5]P) \cdot f_{x,Q}^{p^2}([p^6]P) \cdot f_{x,Q}^{p}([p^7]P) \cdot f_{x,Q}([p^8]P) \big) \\
&\quad \cdot \big(f_{x,P}^{p^8}(Q) \cdot f_{x,[p]P}^{p^7}(Q) \cdot f_{x,[p^2]P}^{p^6}(Q) \cdot f_{x,[p^3]P}^{p^5}(Q) \cdot f_{x,[p^4]P}^{p^4}(Q) \\
&\quad \cdot f_{x,[p^5]P}^{p^3}(Q) \cdot f_{x,[p^6]P}^{p^2}(Q) \cdot f_{x,[p^7]P}^{p}(Q) \cdot f_{x,[p^8]P}(Q) \big)^{-1} \Big]^{p^9-1}
\end{aligned}$$

A.1. Serial computation

Original β-Weil pairing for serial computation are 9 Miller lite functions, 9 full Miller functions, 2 p, 2 p^2, 2 p^3, 2 p^4, 2 p^5, 2 p^6, 2 p^7, 2 p^8, and 1 p^9 Frobenius maps, 18 multiplications, 2 inversions in $\mathbb{F}_{p^{27}}$. From Algorithm 2, the cost of Miller Lite is $34M + 51(125M) + 51(13M + 27M + 216M) + 4(16M + 27M + 216M + 216M) = 21365M$. The cost of Full Miller is $34M + 51(125M) + 51(233M + I + 9M + 189M) + 4(233M + I + 9M + 189M + 189M) = 30870M + 55I$. The total cost is $9 \cdot 21365M + 9(30870M + 55I) + 2(3 \cdot 18M + 6 \cdot 26M) + 18M + 18 \cdot 216M + 2(I + 449M + 62S) = 475463M + 497I$.

Similarly, the computational cost of the original β-Weil pairing on BLS-15 given as $\beta_{15}(P,Q) = \prod_{i=0}^{4} \left(\frac{f_{x,Q}([p^i]P)}{f_{x,[p^i]P}(Q)} \right)^{(p^{4-i})(p^5-1)}$ with $x = -2^{77} - 2^{76} - 2^{68} - 2^{50}$ is $5Miller.Lite + 5Full.Miller + 2(F_1 + F_2 + F_3 + F_4) + 10M_{15} + F_5 + 2I_{15} = 177657M + 401I$, where $Miller.Lite = 14747M$ and $Full.Miller = 20498M + 80I$.

Also, the computational cost of the original β-Weil pairing on BLS-9 given as $\beta_9(P,Q) = \prod_{i=0}^{2} \left(\frac{f_{x,Q}([p^i]P)}{f_{x,[p^i]P}(Q)} \right)^{(p^{2-i})(p^3-1)}$ with $x = -2^{74} - 2^{72} - 2^{46} - 2^{31}$ is $3Miller.Lite + 3Full.Miller + 2(F_1 + F_2) + 6M_9 + F_3 + 2I_9 = 43874M + 232I$, where $Miller.Lite = 6441M$ and $Full.Miller = 7972M + 77I$.

A.2. Parallel computation of β-Weil pairing on BLS-27 using 3 processors

1^{st} processor computes

$$\left(f_{x,Q}^{p^8}(P) \cdot f_{x,Q}^{p^7}([p]P) \cdot f_{x,Q}^{p^6}([p^2]P) \right) \cdot \left(f_{x,P}^{p^8}(Q) \cdot f_{x,[p]P}^{p^7}(Q) \cdot f_{x,[p^2]P}^{p^6}(Q) \right)^{-1}$$

2^{nd} processor computes

$$\left(f_{x,Q}^{p^5}([p^3]P) \cdot f_{x,Q}^{p^4}([p^4]P) \cdot f_{x,Q}^{p^3}([p^5]P) \right) \cdot \left(f_{x,[p^3]P}^{p^5}(Q) \cdot f_{x,[p^4]P}^{p^4}(Q) \cdot f_{x,[p^5]P}^{p^3}(Q) \right)^{-1}$$

3^{rd} processor computes
$\left(f_{x,Q}^{p^2}([p^6]P)\cdot f_{x,Q}^{p}([p^7]P)\cdot f_{x,Q}([p^8]P)\right)\cdot\left(f_{x,[p^6]P}^{p^2}(Q)\cdot f_{x,[p^7]P}^{p}(Q)\cdot f_{x,[p^8]P}(Q)\right)^{-1}$

We see that, 3^{rd} processor's cost is greater than 1^{st} and 2^{nd} processors' cost because 3^{rd} processor require the extra computation of $[p^i]P$ $(1 \leq i \leq 6)$ which does not require in 1^{st} and 2^{nd} processors. Therefore, the computational cost of the 3^{rd} processor is $cost.of.([p^6]P) + 3Miller.Lite + 3Full.MIller + 2(F_1+F_2) + 5M_{27} + I_{27} = 6\cdot 727M + 3(21365M) + 3(30870M + 55I) + 2(26M + 26M) + 5\cdot 216M + (449M + I) = 162700M + 166I$. In the last step, one processor multiplies the three results of each processor and computes the $p^9 - 1$ power at a cost of $3\cdot 216M + 18M + (I + 449M) = 1115M + I$. Therefore, the total cost is $163815M + 167I$.

Appendix B. Computation of the optimal Ate pairing on BLS-27 curve with a new parameter

Remark B.1. The method of [33] (section 3 page 6) for parallelizing the computation of the Miller function $f_{x,Q}$ is the following. We first write $x = 2^w x_1 + x_0$, where $x_0 < 2^w$. We obtain

$$f_{x,Q} = f_{x_1,Q}^{2^w} \cdot f_{2^w,x_1Q} \cdot f_{x_0,Q} \cdot \frac{l_{2^w x_1 Q, x_0 Q}}{V_{xQ}}$$

Thus the computation of $f_{x,Q}$ can be parallelized by computing $f_{x_1,Q}^{2^w}$ on one processor, f_{2^w,x_1Q} on a second processor and $f_{x_0,Q} \cdot \frac{l_{2^w x_1 Q, x_0 Q}}{V_{xQ}}$ on a third processor. This is the case where the three processors are independent. The parameter w should be carefully selected in order to balance the time of the three function computations. And if it is done in the right way we can estimate the cost one processor (the more costly) to be about $\frac{1}{3}$ of the initial Miller loop. Note that we can only parallelize the Miller function but not the final exponentiation.

The optimal Ate pairing on $BLS - 27$ curve in [13] is given by:

$$e_0(Q, P) = f_{x,Q}(P)^{(p^{27}-1)/r}.$$

For $x = -2^{51} - 2^{31} - 2^{21} - 2^8 - 2^4$, the Miller loop executes 51 doubling steps, 4 addition steps, 51 squarings and 59 multiplications in $\mathbb{F}_{p^{27}}$. As in [17], the Miller loop cost of $M27 = 34M + 51(233M + I + 9M) + 4(233M + I + 9M) + 51(125M) + 59 \cdot 189M = 30870M + 55I$.

The final exponentiation is divided into two parts: the easy part $A = f^{p^9-1}$ and the hard part A^d, where $d = (p^{18} + p^9 + 1)/r$.
the easy part is 1 p^9-Frobenius, 1 multiplications and 1 inversion in $\mathbb{F}_{p^{27}}$. That is $18M + 1M_{27} + 1I_{27} = 682M + I$. The element $d = (p^{18}p + ^9 + 1)/r$, is decomposed as
$(x-1)^2 \cdot (p^9 + x^9 + 1) \cdot (p^8 + x.p^7 + ... + x^7.p + x^8) + 3$. The evaluation of the hard part is as follows:

$$A_1 = A^{p^8} \cdot A^{xp^7} \cdot A^{x^2p^6} \cdot A^{x^3p^5} \cdot A^{x^4p^4} \cdot A^{x^5p^3} \cdot A^{x^6p^2} \cdot A^{x^7p} \cdot A^{x^8},$$
$$A_2 = A_1^{(x-1)^2},\ A_3 = A_2^{x^9} \cdot A_2^{p^9} \cdot A_2,\ A_4 = A^3 \cdot A_3.$$

The computation of the hard part requires 17 powers of x, 2 powers of $x-1$, 11 multiplications in $\mathbb{F}_{p^{27}}$ and p, $p^2, p^3, p^4, p^5, p^6, p^7, p^8, p^9-$Frobenius maps. The negative coefficient in the value of x requires 19 inversions in the cyclotomic subgroup when raising to the power of x during the final exponentiation. (Note that $A^{-1} = A^{p^9} \cdot A^{p^{18}}$ and cost $2 \cdot 18M + 1M_{27} = 252M$). The hard part then cost
$17(51S_{27}+4M_{27})+2(51S_{27}+5M_{27})+11M_{27}+2\cdot 18M+6\cdot 26M+19I_{G_{\varphi_3(p^9)}} = 145329M$. The computational cost of the optimal Ate pairing over $BLS-27-$curve is then $(31365M+55I)+(682M+I)+145329M =$ **176881 M+56I**.

Appendix C. Computation of the optimal Ate pairing on BLS-15 curve with a new parameter

Similarly, we evaluate $e_0(Q,P) = f_{x,Q}(P)^{(p^{15}-1)/r}$ with $x = -2^{77}-2^{76}-2^{68}-2^{50}$ on BLS-15 curve. The computational cost of Miller full is $20498M + 80I$ and best cost of the final exponentiation is $9E_x+2E_{x-1}+12M_{15}+S_{15}+I_{15}+3I_{cyc}+3F_1+F_2+F_3+F_4+2F_5+F_6+F_7$ see [34]) for this we will add 11 cyclotomique inversion due to the negative parameter x. $E_x = 77S_{15}+3M_{15}$, $E_{x-1} = 77S_{15}+4M_{15}$ and $I_{cyc} = 54M$. Thus the final exponentiation cost $70971M+I$ and the computational cost of the optimal ate is then $91469M+81I$.

Appendix D. Computation of the optimal Ate pairing on BLS-9 curve with a new parameter

We evaluate $e_0(Q,P) = f_{x,Q}(P)^{(p^9-1)/r}$ with $x = -2^{74}-2^{72}-2^{46}-2^{31}$ on BLS-9 curve. The computational cost of Miller full is $7972M + 77I$ and the optimal cost of the final exponentiation is $5E_x + 2E_{x-1} + 7M_9 + S_9 + I_9 + F_1 + F_2 + 2F_3$ see [34]) for this we will add 7 cyclotomique inversion due to the negative parameter x. $E_x = 74S_9 + 3M_9$, $E_{x-1} = 74S_9 + 4M_9$ and $I_{cyc} = 33M$. Thus the final exponentiation cost $14393M + I$ and the computational cost of the optimal Ate is then $22365M + 78I$.

Azebaze Guimagang Laurian
Department of Mathematics, Faculty of Science,
The University of Yaounde 1, P.O BOX 812 Yaounde, Cameroon

e-mail: azebazelaurian@yahoo.fr

Fouotsa Emmanuel
Department of Mathematics, Higher Teacher Training College,
The University of Bamenda, P.O BOX 39 Bambili, Cameroon

e-mail: emmanuelfouotsa@yahoo.fr

El Mrabet Nadia
Mines Saint-Etienne, CEA-Tech, Departement SAS, F - 13541 Gardanne France
e-mail: nadia.el-mrabet@emse.fr

Pecha Njiahouo Aminatou
Ecole Nationale Suprieure Polytechnique de Maroua, Université de Maroua P.O.Box 46 Maroua, Cameroon
e-mail: aminap2001@yahoo.fr