On the Computation of the Optimal Ate Pairing at the 192-bit Security Level

Loubna Ghammam [1] and Emmanuel Fouotsa[2]

(1) IRMAR, UMR CNRS 6625, Université Rennes 1,
Campus de Beaulieu 35042 Rennes cedex, France.
(1) Laboratoire d'électronique et de microé lectronique
FSM Monastir Université de Monastir
loubna.ghammam@yahoo.fr
(2) LMNO, UMR CNRS 5139 Université de Caen,
Campus 2, 14032 Caen Cedex, France.
(2) Higher Teacher Training College, University of Bamenda,
P.O.Box 39 Bambili, Cameroon
emmanuel.fouotsa@yahoo.fr

Abstract. Barreto, Lynn and Scott elliptic curves of embedding degree 12 denoted BLS12 have been proven to present fastest results on the implementation of pairings at the 192-bit security level [1]. The computation of pairings in general involves the execution of the Miller algorithm and the final exponentiation. In this paper, we improve the complexity of these two steps up to 8% by searching an appropriate parameter. We compute the optimal ate pairing on BLS curves of embedding degree 12 and we also extend the same analysis to BLS curves with embedding degree 24. Furthermore, as many pairing based protocols are implemented on memory constrained devices such as SIM or smart cards, we describe an efficient algorithm for the computation of the final exponentiation less memory intensive with an improvement up to 25% with respect to the previous work.
Keywords: BLS curves, Optimal Ate pairing, final exponentiation, memory resources, Miller loop.

1 Introduction

The performance of pairing-based protocols depends on the efficiency of pairing computation. This computation consists of two main parts: the Miller step and the final exponentiation. Given an elliptic curve E defined over a finite field $\mathbb{F}_p$ and two points R and S on E, the Miller step consists of computing the function $f_{u,R}$ with divisor $\mathrm{Div}(f_{u,R}) = u(R) - ([u]R) - (u-1)(\mathcal{O})$ where u is an integer and $\mathcal{O}$ denotes the identity element of the group of points of the elliptic curve. The efficiency (the number of operations) of the Miller step depends on the bit length

$\log_2(u)$ of u and its Hamming weight since this step uses the *double-and-add* Miller algorithm [2]. The final exponentiation consists of raising the result $f_{u,R}(S)$ of the Miller step to the power of $\frac{p^k-1}{r}$ as follows $f_{u,R}(S)^{\frac{p^k-1}{r}} = \left(f_{u,R}(S)^{\frac{p^k-1}{\phi_k(p)}}\right)^{\frac{\phi_k(p)}{r}}$, where r is a large prime dividing the order of the group of rational points of E, k is called the embedding degree of E and is the smallest integer such that r divides p^k-1; and $\phi_k(x)$ is the k-th cyclotomic polynomial. The computation of the first part of the final exponentiation i.e the computation of $A = f_{u,R}(S)^{\frac{p^k-1}{\phi_k(p)}}$ is generally cheap as it consists of few multiplications, an inversion and taking p-th power in $\mathbb{F}_{p^k}$. The second part which consists of the computation $A^{\frac{\phi_k(p)}{r}}$ is more difficult and is called the hard part. An efficient method to compute this term is described by Scott et *al.* [3]. They suggested to write $d = \frac{\phi_k(p)}{r}$ in base p as $d = d_0 + d_1 p + ... + d_{\phi(k)-1}p^{\phi(k)-1}$ and find a short vectorial addition chain to compute A^d much more efficiently than the naive method.

In this paper, we are interested in the improvement on the computation of the Miller step and the final exponentiation on BLS curves [4] at the 192-bit security level. Indeed, the 192-bit security level is the highest security level for public-key operations in the National Security Agency's Suite B Cryptography standard [5]. Also based on the results concerning implementation of pairings on elliptic curves with embedding degree 12 at the 192-bit security level, BLS12 curves have the fastest performances [1]. Specifically, we search for an adequate value of the parameter u to reduce the number of addition steps in the Miller algorithm and the complexity of the final exponentiation. Furthermore, as many pairing-based protocols are implemented on memory constrained devices, we describe an efficient algorithm for the computation of the final exponentiation with less temporary variables. The improvement in this work is up to 25% in memory resources and about 8% in the complexity of the optimal ate pairing compared to the previous work done in [1]. The rest of the paper is organised as follows: In Section 2 we present the Barreto-Lynn and Scott curves (BLS) and a brief description of optimal pairings. The Section 3 is the state of the art on the computation of the hard part of the final exponentiation. In this section we follow the work of Aranha et al.[1] and we study the number of temporary variables used for the computation of the final exponentiation using their approach in the development of the exponent. In Section 4 we present a new development of the exponent in

the hard part of the final exponentiation for BLS12 curves which enable us to improve the cost of the computation and requires less memory resources comparatively to the results in [1]. We propose also in section 5 a new parameter u of the BLS12 curves which leads to the reduction of the Miller loop and an efficient final exponentiation. The Section 6 presents a similar analysis for BLS24 curves. The results obtained are an improvement up to 8% than Aranha et *al.* results. In Section 7 we compare the results obtained in this work with previous fast results on optimal ate pairings at the 192-bit security level. Section 8 concludes our paper.

Notations:
In this paper we denote by:

- M_k a multiplication in $\mathbb{F}_{p^k}$.
- S_k a squaring in $\mathbb{F}_{p^k}$.
- F_k a Frobenius map application in $\mathbb{F}_{p^k}$.
- I_k an inversion in $\mathbb{F}_{p^k}$.

A multiplication, a squaring and an inversion in $\mathbb{F}_p$ are denoted by M, S and I respectively.

2 Barreto-Lynn-Scott Curves (BLS12) and Optimal Ate Pairings

In 2002, Barreto, Lynn and Scott presented in [4] a method to generate pairing-friendly elliptic curves over a prime field $\mathbb{F}_p$ with embedding degree $k = 12$. BLS12 are defined over $\mathbb{F}_p$ by the following equation:

$$E : y^2 = x^3 + b$$

and by a parameter $u \in \mathbb{Z}$ such that:

$$\begin{cases} p = (u-1)^2(u^4 - u^2 + 1)/3 + u \\ r = u^4 - u^2 + 1 \\ t = u + 1 \end{cases} \tag{1}$$

where t is the trace of the Frobenius map on the curve. The parameter u is chosen such that p and r are prime and have the sizes corresponding to the desired security level following the recommendations in [6]. At the 192-bit security level for BLS12, p and r are of at least 640 and 384 bits in sizes respectively.
The concept of optimal pairing is defined in [7]. Let $\pi_p : E\left(\overline{\mathbb{F}_p}\right) \rightarrow$

$E\left(\overline{\mathbb{F}_p}\right), (x,y) \mapsto (x^p, y^p)$ be the Frobenius endomorphism on the curve where $\overline{\mathbb{F}_p}$ is an algebraic closure of $\mathbb{F}_p$. Denote $[n] : P \longmapsto [n]P$ the endomorphism defined on $E(\mathbb{F}_p)$ which consists of adding P to itself n times. Let $G_1 = E(\mathbb{F}_p)[r]$ be the r-torsion subgroup of $E(\mathbb{F}_p)$. Let $G_2 = E'(\mathbb{F}_{p^2})[r] \cap \mathrm{Ker}(\pi_p - [p])$ where E' is the sextic twist of E and $G_3 = \mu_r$ is the subgroup of $\mathbb{F}^{\star}_{p^{12}}$ consisting of r-th roots of unity. An explicit formula of the optimal ate pairing on BLS12 curves is detailed in [1] and is given in Proposition 1.

Proposition 1 *[7] The optimal ate pairing over the parametrized BLS12 curves is the bilinear and non degenerated map:*

$$\begin{aligned} e_{opt} : G_1 \times G_2 &\rightarrow G_3 \\ (P,Q) &\longmapsto f_{u,Q}(P)^{\frac{p^{12}-1}{r}} \end{aligned}$$

where $f_{u,R}$ is the function with divisor $\mathrm{Div}(f_{u,R}) = u(R) - ([u]R) - (u-1)(\mathcal{O})$

Let $u = u_n 2^n + ... + u_1 2 + u_0$ with $u_i \in \{-1, 0, 1\}$. Let $\ell_{R,Q}$ be the line passing through the points R and Q of the elliptic curves.

The function $f_{u,Q}$ (and in general the pairing $f_{u,Q}(P)^{\frac{p^{12}-1}{r}}$) is efficiently computed thanks to the following algorithm known as the Miller's algorithm [2].

Miller algorithm and pairing computation:
Input: u, $n = \log_2(u)$, P,Q
Output:$f_{u,Q}(P)^{\frac{p^{12}-1}{r}}$

1: Set $f_1 \leftarrow 1$ and $R \leftarrow Q$
2: **For** $i = n-1$ **down to** 0 **do**
3: $f_1 \leftarrow f_1^2 \cdot \ell_{R,R}(P)$, $R \leftarrow 2R$ Doubling step 1
5: **if** $u_i = 1$ **then**
6: $f_1 \leftarrow f_1 \cdot \ell_{R,Q}(P)$ $R \leftarrow R+Q$, **end if** Addition step
7: **if** $u_i = -1$ **then**
8: $f_1 \leftarrow f_1 \cdot \ell_{R,-Q}(P)$ $R \leftarrow R-Q$, **end for** Addition step
10: **return** $e = f_1^{\frac{p^{12}-1}{r}}$ Final exponentiation

Remark 2 *The loop length of the Miller algorithm is $log_2(u)$ and the addition steps are done only if $u_i \in \{-1, 1\}$. Therefore any u with a smaller bit size (l_u) and low Hamming weight (w_u) will be a good solution for the efficiency of the algorithm. Example 3 gives an illustration.*

Example 3 *Aranha et al. presented in [1] a value u for the BLS12 curve which is a 107-bit integer length ($l_u = 108$) of Hamming weight $w_u = 4$:*

$$u = -2^{107} + 2^{105} + 2^{93} + 2^5$$

This parameter yields a 638-bit prime p and 427-bit prime r. For this parameter u, in Miller loop, the number of doubling steps is 107 *and the number of additions steps is* 4*.*

3 Previous Work on the Computation of Final Exponentiation on BLS12

The computation of optimal ate pairing on BLS12 curves is done in [1]. However the authors do not take into consideration the number of temporary variables involved especially in the computation of the final exponentiation. This may be a drawback when implementing pairings over memory constrained devices. In this section we try to overcome this drawback by adding details to their computation, especially for the hard part of the final exponentiation given by:

$$\frac{p^{12}-1}{r} = \left(p^6-1\right)\left(p^2+1\right)\frac{p^4-p^2+1}{r}$$

To compute the first part $f = f_1^{\left(p^6-1\right)\left(p^2+1\right)}$ we have to perform just two easy Frobenius operations, two multiplications and an inversion in $\mathbb{F}_{p^{12}}$. This inversion is a hard operation, however it has an important consequence for the rest of the computation. Indeed, powering f_1 to the p^6-1 makes the result unitary [8]. By this way, during the hard part of the final exponentiation (the computation of $f^{\frac{p^4-p^2+1}{r}}$) all the elements involved are unitary. This simplifies the computations. For example any future inversion can be implemented as a Frobenius operator, more precisely $f^{-1} = f^{p^6}$ which is just a conjugation [8], [9]. Consequently, we assume in this section that inversions are free. The exponent $\frac{p^4-p^2+1}{r}$ of the hard part can be simply written as a polynomial in p of degree 3:

$$\frac{p^4-p^2+1}{r} = \lambda_0 + \lambda_1 p + \lambda_2 p^2 + \lambda_3 p^3$$

where

$$\begin{cases} \lambda_0 = u^5 - 2u^4 + 2u^2 - u + 3 \\ \lambda_1 = u^4 - 2u^3 + 2u - 1 \\ \lambda_2 = u^3 - 2u^2 + u \\ \lambda_3 = u^2 - 2u + 1 \end{cases} \tag{2}$$

In [1], the computation of $f^{\lambda_0+\lambda_1 p+\lambda_2 p^2+\lambda_3 p^3}$ is done in 2 steps: First they compute

$$f \longrightarrow f^{-2} \longrightarrow f^{u} \longrightarrow f^{2u} \longrightarrow f^{u-2} \longrightarrow f^{u^2-2u} \longrightarrow$$

$$f^{u^3-2u^2} \longrightarrow f^{u^4-2u^3} \longrightarrow f^{u^4-2u^3+2u} \longrightarrow f^{u^5-2u^4+2u^2}$$

which requires 5 exponentiations by u, 2 multiplications in $\mathbb{F}_{p^{12}}$ and 2 cyclotomic squarings. The second step is applying Frobenius maps and multiplying terms together to have the following expression:

$$f^d = f^{u^5-2u^4+2u^2}\left(f^{u-2}\right)^{-1}\left(f^{u^4-2u^3+2u}f^{-1}\right)^{p}\left(f^{u^3-2u^2}f^{u}\right)^{p^2}\left(f^{u^2-2u}f\right)^{p^3}$$

which requires 3 Frobenius maps and also other 8 multiplications. Therefore, the total cost of the hard part of the final exponentiation through Aranha et *al.* method is 5 exponentiations by u, 10 multiplications, 2 squarings and 3 Frobenius in $\mathbb{F}_{p^{12}}$. In their paper, Aranha et al. do not specify the number of temporary variables used to compute f^d. This can be a requirement if we think about implementation of pairings in a restricted environment. That's why, we have computed them and have found that we need to use at least 6 temporary variables in $\mathbb{F}_{p^{12}}$ to compute the hard part of the final exponentiation as shown in Algorithm 1. A magma code to check the correctness of this algorithm is available here. The overall cost of computing f^d is then $(5l_u - 3)S_{12} + (5w_u + 5)M_{12} + 3F_{12}$. Considering the value of u chosen in example 3, the total cost of this algorithm is $537S_{12} + 25M_{12} + 3F_{12}$.

Remark 4 *In fact, neither Aranha et al [1] nor the present authors compute the optimal ate pairing itself, but rather its cube. The advantage of that is that the coefficient λ_i become integers.*

4 New development of d with u proposed in [1]

As we earlier said in the introduction, our aim in this paper is to reduce the complexity and the number of temporary variables used to compute the hard part of the final exponentiation $f^{\frac{p^4-p^2+1}{r}}$. In this Section we propose another development of the exponent d which enable us to use less temporary variables and therefore decrease the number of multiplications in $\mathbb{F}_p$. Recall that $d = \frac{p^4-p^2+1}{r} = \lambda_0 + \lambda_1 p + \lambda_2 p^2 + \lambda_3 p^3$. To improve the cost of the computations we rewrite λ_i with $0 \leq i \leq 3$ differently as

Algorithm 1: Aranha et *al.*[1] development	Computed Terms and comments	Cost
Input: $f,\ u$		
Output: $f^{\frac{p^4-p^2+1}{r}}$		
Temp. var.: t_0, t_1, t_2, t_3, t_4		
t_5		
$t_0 \leftarrow f^{-2}$	f^{-2}	S_{12}
$t_5 \leftarrow f^u$		$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_1 \leftarrow t_5^2$	f^{2u}	S_{12}
$t_3 \leftarrow t_0 t_5$	f^{u-2}	M_{12}
$t_0 \leftarrow t_3^u$	f^{u^2-2u}	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_2 \leftarrow t_0^u$	$f^{u^3-2u^2}$	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_4 \leftarrow t_2^u$	$f^{u^4-2u^3}$	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_4 \leftarrow t_1 t_4$		M_{12}
$t_1 \leftarrow t_4^u$	$f^{u^5-2u^4+2u^2}$	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_3 \leftarrow t_3^{-1}$		
$t_1 \leftarrow t_3 t_1$		M_{12}
$t_1 \leftarrow t_1 f$	f^{λ_0}	M_{12}
$t_3 \leftarrow f^{-1}$		
$t_0 \leftarrow t_0 f$	f^{λ_3}	M_{12}
$t_0 \leftarrow t_0^{p^3}$		F_{12}
$t_4 \leftarrow t_3 t_4$	f^{λ_1}	M_{12}
$t_4 \leftarrow t_4^p$		F_{12}
$t_5 \leftarrow t_2 t_5$	f^{λ_2}	M_{12}
$t_5 \leftarrow t_5^{p^2}$		F_{12}
$t_5 \leftarrow t_5 t_0$		M_{12}
$t_5 \leftarrow t_5 t_4$		M_{12}
$t_5 \leftarrow t_5 t_1$	$f^{\frac{p^4-p^2+1}{r}}$	M_{12}
return t_5		

Table 1. Temporary variables used in the previous work [1]

follows:

$$\begin{cases} \lambda_0 = \lambda_1 u + 3 \\ \lambda_1 = \lambda_2 u - \lambda_3 \\ \lambda_2 = \lambda_3 u \\ \lambda_3 = u^2 - 2u + 1 \end{cases} \tag{3}$$

From these new relations satisfied by λ_0, λ_1, λ_2 and λ_3 we get algorithm 2 in Table 2 which allows us to compute $f^{\frac{p^4-p^2+1}{r}}$. A magma code to check the correctness of this algorithm is available here.

To compute any exponentiation, we use the *square and multiply* algorithm [10]. The cost of the four exponentiations by u in this algorithm is $4(l_u-1)S_{12}+4(w_u-1)M_{12}$ and the cost of the exponentiation by $u/2$ is

Algorithm 2: new variant of Aranha et *al.*[1]	Computed Terms and comments	Cost
Input: $f,\ u$		
Output: $f^{\frac{p^4-p^2+1}{r}}$		
Temp. var.: t_0, t_1, t_2, t_3, t_4		
$t_0 \leftarrow f^2$		S_{12}
$t_1 \leftarrow t_0^u$		$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_2 \leftarrow t_1^{u/2}$	f^{u^2}	$(l_u-2)S_{12}+(w_u-1)M_{12}$
$t_3 \leftarrow f^{-1}$		
$t_1 \leftarrow t_3 t_1$	f^{2u-1}	M_{12}
$t_1 \leftarrow t_1^{-1}$	f^{-2u+1}	
$t_1 \leftarrow t_1 t_2$	f^{λ_3}	M_{12}
$t_2 \leftarrow t_1^u$	f^{λ_2}	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_3 \leftarrow (t_2)^u$	$f^{\lambda_2 u}$	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_1 \leftarrow t_1^{-1}$	$f^{-\lambda_3}$	
$t_3 \leftarrow t_1 t_3$	f^{λ_1}	M_{12}
$t_1 \leftarrow t_1^{-1}$	f^{λ_3}	
$t_1 \leftarrow t_1^{p^3}$	$f^{\lambda_3 p^3}$	F_{12}
$t_2 \leftarrow t_2^{p^2}$	$f^{\lambda_2 p^2}$	F_{12}
$t_1 \leftarrow t_1 t_2$	$f^{\lambda_3 p^3} f^{\lambda_2 p^2}$	M_{12}
$t_2 \leftarrow t_3^u$	$f^{\lambda_1 u}$	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_2 \leftarrow t_2 t_0$		M_{12}
$t_2 \leftarrow t_2 f$	f^{λ_0}	M_{12}
$t_1 \leftarrow t_1 t_2$	$f^{\lambda_3 p^3} f^{\lambda_2 p^2} f^{\lambda_0}$	M_{12}
$t_2 \leftarrow t_3^p$	$f^{\lambda_1 p}$	F_{12}
$t_1 \leftarrow t_1 t_2$	$f^{\lambda_3 p^3} f^{\lambda_2 p^2} f^{\lambda_1 p} f^{\lambda_0}$	M_{12}
return t_1		

Table 2. Temporary variables used with the new development of d

$(l_u-2)S_{12}+(w_u-1)M_{12}$. The overall cost of computing f^d with steps given in Algorithm 2 is then $4(l_u-1)S_{12}+(l_u-2)S_{12}+S_{12}+(5w_u+3)M_{12}+3F_{12}$. We summarise in Table 3 the two results from Table 1 and Table 2.

Method	Complexity			Temp. var.
	S_{12}	M_{12}	F_{12}	
Aranha et *al.*[1] (algorithm 1)	$5l_u-3$	$5w_u+5$	3	6
This work (algorithm 2)	$5l_u-5$	$5w_u+3$	3	4

Table 3. Comparison between Aranha et al.[1] and our new development

Through Table 3 we remark that our approach gives faster results than the method given in [1] for the computation of the hard part of the final exponentiation. We saved 2 squarings and 2 multiplications in $\mathbb{F}_{p^{12}}$ thanks to the fact that u is even. We have also decreased the used memory resources, we have used 4 temporary variables instead of 6 in [1]. In order to give a more explicit comparison we consider Example 5.

Example 5 *Let E a BLS12 elliptic curve defined over a prime field $\mathbb{F}_p$ by*

$$E : y^2 = x^3 + 4$$

Based on the parameter $u = -2^{107} + 2^{105} + 2^{93} + 2^5$ proposed by Aranha et al. *an exponentiation by u needs 3 multiplications and 107 squarings in $\mathbb{F}_{p^{12}}$. A detailed comparison is given in the following Table.*

Method	Complexity			Temp. var.
	S_{12}	M_{12}	F_{12}	
Aranha et *al.*[1] (algorithm 1)	537	25	3	6
This work (algorithm 2)	535	23	3	4

Table 4. Comparison between Aranha et al.[1] and our new development

For a full comparison, we give the complexity of each $\mathbb{F}_{p^{12}}$ operation in $\mathbb{F}_p$. In our case -1 is not a square and $(1+\alpha)$ is neither a cube nor a square.
The field $\mathbb{F}_{p^{12}}$ is built using the following extension tower.

- $\mathbb{F}_{p^2} = \mathbb{F}_p[\alpha]/(\alpha^2 + 1)$
- $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[\beta]/\left(\beta^3 - (\alpha + 1)\right)\big)$
- $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[\gamma]/\left(\gamma^2 - \beta\right)$

The cost of arithmetic operations in $\mathbb{F}_p$, $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{12}}$ are detailed in [11], [12], [1].

5 Development of d with a New Parameter u

Our aim is to reduce the complexity of the computation of the pairing as much as possible for both the computation of the hard part of the final exponentiation and also the Miller loop. We wrote a Pari/GP code to find

a suitable u with low hamming weight and minimal number of bits for the 192- bits security level.

Proposition 6 *The best value of u we were able to find is*

$$u = -2^{107} + 2^{84} + 2^{19}$$

which gives p a 641-bit prime number and r a 428-bit prime number. The Hamming weight of u is $w_u = 3$, this low Hamming weight has an advantage because any exponentiation by u needs only 2 multiplications instead of 3 needed if we use the parameter proposed by Aranha et al.

Now we present the improved cost of our development of the hard part of the final exponentiation using the new value of the parameter u in the following Table 5. We can deduce that using the new parameter u in our

Method	Complexity			Number of Temp. var. must used
	S_{12}	M_{12}	F_{12}	
Aranha et *al.*[1]	537	25	3	6
This work with u of Aranha et *al.*[1]	535	23	3	4
This work with new u	535	18	3	4

Table 5. Comparison of the cost of the hard part of the final exponentiation

development is a fast alternative for computing the hard part of the final exponentiation.

The overall cost of the final exponentiation $f^{(p^6-1)(p^2+1)\frac{p^4-p^2+1}{r}}$ is an inversion in $\mathbb{F}_{p^{12}}$, 10 multiplications, 4 Frobenius, one cyclotomic squaring, 5 exponentiations by u and 1 exponentiation by $u/2$, where the easiest part costs 2 multiplications, an inversion and a Frobenius. Any exponentiation by our new parameter u requires 107 compressed squarings, simultaneous decompression of 4 field elements when Karabina's exponentiation technique [13] is employed and 2 multiplications in $\mathbb{F}_{p^{12}}$. So we have to perform $107(6S_2)+4(3M_2+3S_2)+3(3M_2)+I_2+2(18M_2) = (57M_2+654S_2+I_2)$ to compute any exponentiation by u. The overall cost of the final exponentiation is therefore $4(57M_2 + 654S_2 + I_2) + (57M_2 + 648S_2 + I_2) + 4(15M) + 10(18M_2) + (23M_2 + 11S_2 + I_2) + 9S_2$=8116M+6I.
As computed in [1], the cost of a multiplication in $\mathbb{F}_{p^{12}}$ is about $54M$, a cyclotomic squaring costs $18M$ and a Frobenius in $\mathbb{F}_{p^{12}}$ is 15 multiplications in $\mathbb{F}_p$. Consequently, the cost of the final exponentiation using the

new parameter u and our new development is less than the cost given in [1]. We saved about 408 multiplications in $\mathbb{F}_p$ which is about **5%** of the overall cost of the final exponentiation. The advantage of our parameter u is also that we reduced the computational cost in the Miller loop. The number of doubling step in Miller algorithm is determined by the length of u in base 2 which is $\lfloor log_2(u) \rfloor = l_u$. The Hamming weight of u determines the number of addition steps in Miller's algorithm.
That's why, using the new parameter u that we proposed in this paper we have to perform just 2 addition steps instead of 3 done in [1]. Therefore we save an addition step with line evaluation and also a multiplication in $\mathbb{F}_{p^{12}}$. This gain is about 80 multiplications in $\mathbb{F}_p$ which represents **1%**.

6 Optimal Ate Pairings over BLS24 Curves

Although BLS12 curves present the fastest results for the implementation of pairings at the 192 bits security level [1], BLS curves of embedding degree 24 are also well suited for implementing pairings at the high security level [14]. The objective of this section is to improve the cost of the computation of the optimal ate pairing over BLS24 curves. The analysis follows the same approach we used in the case of BLS12 curves. Mainly, a new parameter is obtained with low hamming weight. This enables us to improve the cost of the Miller loop and the computation of the final exponentiation. BLS24 curves are families of elliptic curves parametrized as follows:

$$\begin{cases} p = (u-1)^2(u^8 - u^4 + 1)/3 + u \\ r = u^8 - u^4 + 1 \\ t = u + 1 \end{cases} \tag{4}$$

The authors in [1] consider the implementation of optimal ate pairing on the BLS24 curve defined by the equation $y^2 = x^3 + 4$ and with the parameter $u = -2^{48} + 2^{45} + 2^{31} - 2^7$.

6.1 Previous results on BLS24 curves

The final exponentiation for BLS24 curves is

$$\frac{p^{24} - 1}{r} = \left(p^{12} - 1\right)\left(p^4 + 1\right)\frac{p^8 - p^4 + 1}{r}$$

The exponent $\frac{p^8-p^4+1}{r}$ of the hard part of the final exponentiation is written as

$$\frac{p^8-p^4+1}{r} = \sum_{i=0}^{\phi(24)-1} \lambda_i p^i = \lambda_0 + \lambda_1 p + \lambda_2 p^2 + \cdots + \lambda_7 p^7$$

where

$$\begin{cases} \lambda_0 = u^9 - 2u^8 + u^7 - u^5 + 2u^4 - u^3 + 3 \\ \lambda_1 = u^8 - 2u^7 + u^6 - u^4 + 2u^3 - u^2 \\ \lambda_2 = u^7 - 2u^6 + u^5 - u^3 + 2u^2 - u \\ \lambda_3 = u^6 - 2u^5 + u^4 - u^2 + 2u - 1 \\ \lambda_4 = u^5 - 2u^4 + u^3 \\ \lambda_5 = u^4 - 2u^3 + u^2 \\ \lambda_6 = u^3 - 2u^2 + u \\ \lambda_7 = u^2 - 2u + 1 \end{cases} \tag{5}$$

The hard part of the final exponentiation is computed as

$$f^d = f^{\lambda_0} f^{\lambda_1 p} f^{\lambda_2 p^2} f^{\lambda_3 p^3} f^{\lambda_4 p^4} f^{\lambda_5 p^5} f^{\lambda_6 p^6} f^{\lambda_7 p^7}$$

Following [1], the computation of f^d needs 9 exponentiations by u, 7 Frobenius operations, 2 cyclotomic squarings and 12 multiplications in $\mathbb{F}_{p^{24}}$.
Considering the parameter $u = -2^{48} + 2^{45} + 2^{31} - 2^7$, an exponentiation by u requires $(l_u - 1)$ squarings and $(w_u - 1)$ multiplications in $\mathbb{F}_{p^{24}}$. The cost of the hard part of the final exponentiation presented in [1] is then $(9(l_u - 1) + 2)\, S_{24}$, $(9(w_u - 1) + 12)\, M_{24}$ and 7 Frobenius operations.

6.2 Improvement of the cost of Optimal Ate pairing on BLS24 curves

To improve the computation of f^d we observed that the coefficients in the decomposition of d verify the following relations:

$$\begin{cases} \lambda_0 = \lambda_1 u + 3 \\ \lambda_1 = \lambda_2 u \\ \lambda_2 = \lambda_3 u \\ \lambda_3 = \lambda_4 u - \lambda_7 \\ \lambda_4 = \lambda_5 u \\ \lambda_5 = \lambda_6 u \\ \lambda_6 = \lambda_7 u \\ \lambda_7 = u^2 - 2u + 1 \end{cases} \tag{6}$$

Using these relations we can evaluate f^d in Algorithm 3.

Algorithm 3: BLS24 curves.	Computed Terms and comments	Cost
Input: $f,\ u$		
Output: $f^{\frac{p^8-p^4+1}{r}}$		
Temp. var.: $t_0, t_1, t_2, t3, t4$		
$t_7 \leftarrow f^2$		S_{24}
$t_1 \leftarrow t_7^u$	f^{2u}	$(l_u-1)S_{24} + (w_u-1)M_{24}$
$t_2 \leftarrow t_1^{u/2}$	f^u	$(l_u-2)S_{24} + (w_u-1)M_{24}$
$t_3 \leftarrow t_2^{-1}$		
$t_2 \leftarrow t_1 t_3$		M_{24}
$t_2 \leftarrow t_2 f$	f^{λ_7}	M_{24}
$t_3 \leftarrow t_2^u$	f^{λ_6}	$(l_u-1)S_{24} + (w_u-1)M_{24}$
$t_4 \leftarrow t_3^u$	f^{λ_5}	$(l_u-1)S_{24} + (w_u-1)M_{24}$
$t_3 \leftarrow t_3^{p^6}$	$f^{\lambda_6 p^6}$	F_{24}
$t_4 \leftarrow t_4^{p^5}$	$f^{\lambda_5 p^5}$	F_{24}
$t_3 \leftarrow t_3 t_4$		M_{24}
$t_5 \leftarrow t_4^u$	f^{λ_4}	$(l_u-2)S_{24} + (w_u-1)M_{24}$
$t_6 \leftarrow t_5^u$		$(l_u-1)S_{24} + (w_u-1)M_{24}$
$t_0 \leftarrow t_2^{-1}$		
$t_6 \leftarrow t_6 t_0$	f^{λ_3}	M_{24}
$t_5 \leftarrow t_6^{p^3}$	$f^{\lambda_3 p^3}$	F_{24}
$t_3 \leftarrow t_3 t_5$		M_{24}
$t_5 \leftarrow t_6^u$	f^{λ_2}	$(l_u-1)S_{24} + (w_u-1)M_{24}$
$t_0 \leftarrow t_5^{p^2}$	$f^{\lambda_2 p^2}$	F_{24}
$t_3 \leftarrow t_3 t_0$		M_{24}
$t_6 \leftarrow t_5^u$	f^{λ_1}	$(l_u-1)S_{24} + (w_u-1)M_{24}$
$t_0 \leftarrow t_6^p$	$f^{\lambda_1 p}$	F_{24}
$t_3 \leftarrow t_3 t_0$		M_{24}
$t_5 \leftarrow t_6^u$	f^{λ_0}	$(l_u-1)S_{24} + (w_u-1)M_{24}$
$t_2 \leftarrow t_2^{p^7}$	$f^{\lambda_7 p^7}$	F_{24}
$t_5 \leftarrow t_5 t_7$		M_{24}
$t_3 \leftarrow t_3 t_2$		M_{24}
$t_3 \leftarrow t_3 t_5$		M_{24}
return t_3		

This algorithm requires $(8(l_u-1))\,S_{24}$, $(l_u-2)\,S_{24}$, S_{24}, $(9(w_u-1)+12)\,M_{24}$ and 7 Frobenius operations in $\mathbb{F}_{p^{24}}$. Thanks to the fact that u is even, we saved 2 squarings in $\mathbb{F}_{p^{24}}$.
As in section 5, we also tried in this case to find a new parameter u which has a low Hamming weight. Our Pari/GP code let us find the following parameter

$$u' = 2^{48} - 2^{30} + 2^{26}$$

which gives p a 479-bit prime number and r a 384-bit prime number. This new parameter u' is a 48−bit integer as the u proposed in [1]. However its

Hamming weight is 3 instead of 4 in the case of [1]. This is an advantage because we have in our case less operations to perform. For the parameters p et r, the extension field $\mathbb{F}_{p^{24}}$ is built using the following tower of extensions:

- $\mathbb{F}_{p^2} = \mathbb{F}_p[\alpha]/(\alpha^2 + 1)$
- $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[\beta]/\left(\beta^3 - (\alpha + 2)\right))$
- $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[\gamma]/\left(\gamma^2 - \beta\right)$
- $\mathbb{F}_{p^{24}} = \mathbb{F}_{p^{12}}[\theta]/\left(\theta^2 - \gamma\right)$

The arithmetic for this tower of extension is presented in [1]. Using the new parameter u', any exponentiation by this parameter costs $(l_u - 1)$ squarings and $(w_u - 2)$ multiplications in $\mathbb{F}_{p^{24}}$. Because the Hamming weight of u' is 3, this enables to save one multiplication in $\mathbb{F}_{p^{24}}$ in each exponentiation by u' giving a total of 9 saved multiplications in $\mathbb{F}_{p^{24}}$. Therefore the hard part of the final exponentiation requires $(8(l_u - 1))\, S_{24}$, $(l_u - 2)\, S_{24}$, S_{24}, $(9(w_u - 2) + 12)\, M_{24}$ and 7 Frobenius operations. Then the overall cost of the computation of the final exponentiation is $8\,(48(12M_2) + 89M_2 + 2S_2 + 2(54M_2) + I_2) + 8(45M) + 14(54M_2) + 18M_2 + (47(12M_2) + 89M_2 + 2S_2 + 2(54M_2) + I_2) = 7802M_2 + 29S_2 + 400M + 10I$. In term of the computation of the hard part of the final exponentiation, our method is faster than Aranha et al. method presented in [1]. We saved 1548 multiplications in $\mathbb{F}_p$ which is about **8%**.

The fastest cost of the Miller loop for computing optimal ate pairing over BLS24 curves is reported in [1]. The doubling step costs $21M_2 + 8M$ for the point doubling and $36M_2$ for updating the Miller function in this step. The addition step costs $37M_2 + 8M$ for the point addition and $39M_2$ for updating the Miller function in this step. In this work, we presented a new parameter u' which enable us to perfom only 3 point additions with line evaluations instead of 4 by using u. We win also one multiplication in $\mathbb{F}_{p^{24}}$ which represents 353 multiplications in $\mathbb{F}_p$. Hence the gain is **2.5%**.

7 Comparison

In this paper we were not only interested in the complexity of the optimal ate pairing curves but also on memory usage on BLS12 . We also studied the complexity of the computation of the optimal ate pairing on BLS24 curves using a new development of the hard part of the final exponentiation and we presented a new parameter u. A full comparison of the results in this work with previous fast results on optimal ate pairings at the 192-bit security level is given in the following table.

Curves	Method u	Complexity of Miller loop	Complexity of the final expo
BLS12 Curves	Aranha et *al.*[1]	10865	8524M+6I
	This work	**10785**	**8166M+6I**
BLS24 Curves	Aranha et *al.*[1]	14927	25412M+10I
	This work	**14574**	**23864M+10I**
BN Curves	Aranha et *al.*[1]	16553M	7218M+4I
KSS18 Curves	Aranha et *al.*[1]	13168M	23821M+8I

Table 6. Comparison of previous fast results with this work on pairing at the 192-bit security level

Table 6 shows that our new approach is more efficient than the method presented by Aranha et *al.*[1] in the case of BLS12 curves and also BLS24 curves.

8 Conclusion

For the 192-bit level security, it is recommended to use BLS12 curves because the computation of pairings over this category of curves is more efficient than others curves such as BN curves [15], KSS16 curves [16]. In this paper we improved the computation of the hard part of the final exponentiation and also the computation of Miller loop compared to the costs presented in [1] in BLS12 and BLS24 curves. We implemented our new algorithms in Magma to verify their correctness [17]. As a conclusion, our new methods for computing the hard part of the final exponentiation are more efficient than previous methods in the literature and they are always less memory intensive. Hence, there are an interesting alternative for pairing implementation in restricted environments for the 192-security level.

Acknowledgements. The authors thank Sylvain Duquesne and John Boxall for helpful discussions and comments on this paper.

References

1. Diego F. Aranha, Laura Fuentes-Castañeda, Edward Knapp, Alfred Menezes, and Francisco Rodríguez-Henríquez. Implementing pairings at the 192-bit security level. In *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, pages 177–195, 2012.
2. Victor S. Miller. The weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004.
3. Michael Scott, Naomi Benger, Manuel Charlemagne, Luis J. Dominguez Perez, and Ezekiel J. Kachisa. On the final exponentiation for calculating pairings on ordinary

elliptic curves. In *Pairing-Based Cryptography - Pairing 2009, Third International Conference, Palo Alto, CA, USA, August 12-14, 2009, Proceedings*, pages 78–88, 2009.
4. Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, pages 257–267, 2002.
5. NSA Suite B Cryptography. https://www.nsa.gov/ia/programs/ suitebcryptography/index.shtml.
6. National Institute of Standards and Technology. http://csrc.nist.gov/publications/PubsSPs.html.
7. Frederik Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461, 2010.
8. Michael Scott and Paulo S. L. M. Barreto. Compressed pairings. In *Advances in cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Comput. Sci.*, pages 140–156. Springer, Berlin, 2004.
9. Martijn Stam and Arjen K. Lenstra. Efficient subgroup exponentiation in quadratic and sixth degree extensions. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 318–332, 2002.
10. Sylvain Duquesne and Loubna Ghammam. Memory-saving computation of the pairing final exponentiation on BN curves. *IACR Cryptology ePrint Archive*, 2015:192, 2015.
11. C. C. F. Pereira Geovandro, Marcos A. Simplício Jr., Michael Naehrig, and Paulo S. L. M. Barreto. A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software*, 84(8):1319–1326, 2011.
12. Diego F. Aranha, Koray Karabina, Patrick Longa, Catherine H. Gebotys, and Julio López. Faster explicit formulas for computing pairings over ordinary curves. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 48–68, 2011.
13. Koray Karabina. Squaring in cyclotomic subgroups. *Math. Comput.*, 82(281), 2013.
14. Craig Costello, Kristin E. Lauter, and Michael Naehrig. Attractive subfamilies of BLS curves for implementing high-security pairings. In *Progress in Cryptology - INDOCRYPT 2011 - 12th International Conference on Cryptology in India, Chennai, India, December 11-14, 2011. Proceedings*, pages 320–342, 2011.
15. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*, pages 319–331, 2005.
16. Ezekiel J. Kachisa, Edward F. Schaefer, and Michael Scott. Constructing brezing-weng pairing friendly elliptic curves using elements in the cyclotomic field. *IACR Cryptology ePrint Archive*, 2007:452, 2007.
17. L. Ghammam and E. Fouotsa. http://www.camercrypt.org/BLSformulas.