Faster Final Exponentiation on the KSS18 Curve

Shiping Cai[1], Zhi Hu[2], and ⌧Chang-An Zhao[1,3]

[1] School of Mathematics, Sun Yat-sen University, Guangzhou 510275, P.R. China
E-mail: zhaochan3@mail.sysu.edu.cn
[2] School of Mathematics and Statistics, Central South University, P.R. China
[3] Guangdong Key Laboratory of Information Security, Guangzhou 510006, P.R. China

Abstract. The final exponentiation affects the efficiency of pairing computations especially on pairing-friendly curves with high embedding degree. We propose an efficient method for computing the hard part of the final exponentiation on the KSS18 curve at 192-bit security level. Implementations indicate that the computation of the final exponentiation can be 8.74% faster than the previously fastest result.

Keywords: Final exponentiation, pairings, KSS curves, high security levels

1 Introduction

Pairings play a crucial role in the designs of various cryptographic schemes [9]. The efficiency of these cryptographic schemes heavily depends on the speed of pairing computations. The implementation of pairings consists of two parts: the Miller loop and the final exponentiation. Moreover, as the embedding degree gets large, the final exponentiation becomes very costly compared with the Miller loop. Therefore, it is very meaningful to speed up the final exponentiation.

In order to compute the final exponentiation efficiently, there are several methods such as the vectorial addition (-subtraction) chain [10, 5], the latticed-based method [4], and the method based on cyclotomic polynomials [12, 6] which is beneficial for the BLS family. The method via cyclotomic polynomials is the latest work focus on the final exponentiation that achieves a good performance on the BLS family. It simplifies the process of computations at the final exponentiation step. However, it does not work well on the KSS18 curve which is considered as one of the most efficient curve at 192-bit security level [3, 2], and this work aims to speed up the final exponentiation on this curve. Until now, the state-of-the-art method for the computation of the final exponentiation on the KSS18 curve is the lattice-based method combined with a short vectorial addition-subtraction chain, which requires 7 exponentiations by x, 53 multiplications, 6 squarings and 35 Frobenius operations.

Our Contribution. For accelerating the computation of the final exponentiation on the KSS18 curve, this work uses a trick to save several multiplications compared with the previous fastest method [5]. Besides, the comparison between our new method and the current state-of-the-art is implemented on a 64-bit personal computer, which shows that the computation of the final exponentiation can be 8.74% faster than the previous one.

Organization. An overview of the final exponentiation is introduced in Section 2. In Section 3, we propose a new trick for speeding up the final exponentiation on the KSS18 curve. Section 4 gives the implementation results and concludes this work.

2 Preliminaries

Let $E/\mathbb{F}_p$ be an elliptic curve, where p is a prime. Choose a large prime r such that $r|\#E(\mathbb{F}_p)$ and $\gcd(r,p)=1$. Let k be the embedding degree of $E(\mathbb{F}_p)$ such that k is the smallest integer satisfies $r|p^k-1$ and $r^2 \nmid p^k-1$. Let π_p denote the Frobenius endomorphism over $\mathbb{F}_p$. We use $E[r]$ to denote the $r-$torsion group on E. Here, we choose to compute the Optimal Ate pairing [13, 11]. Therefore, we define two subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$ below:

$$\mathbb{G}_1 = E[r]\bigcap Ker(\pi_p-[1]),\ \mathbb{G}_2 = E[r]\bigcap Ker(\pi_p-[p]).$$

We use μ_r to denote the subgroup of $r-$th roots of unity in $\mathbb{F}^*_{p^k}$.

Choose $P\in\mathbb{G}_1$ and $Q\in\mathbb{G}_2$. Assume that the length of the Miller loop is m. The value of the Miller function $f_{m,Q}(P)$ is computed at the Miller loop step. Then at the final exponentiation step, we raise $f_{m,Q}(P)$ to the power of $e=(p^k-1)/r$ for obtaining a unique value in μ_r. Here, we abbreviate $f_{m,Q}(P)$ to f.

The final exponentiation is generally divided into the easy part and the hard part [10], which means that the exponent e can be written as

$$e=\frac{p^k-1}{r}=\frac{p^k-1}{\phi_k(p)}\cdot\frac{\phi_k(p)}{r}.$$

The first part involves mapping f to $f'=f^{\frac{p^k-1}{\phi_k(p)}}\in\mathbb{G}_{\phi_k(p)}$, where $\mathbb{G}_{\phi_k(p)}$ denotes the $n-$th cyclotomic subgroup of $\mathbb{F}^*_{p^k}$. The hard part involves raising f' to the power of $(\phi_k(p))/r$. The computation of the easy part is relatively inexpensive since it only takes a few multiplications, cheap $p-$power Frobenius operations and an inversion. Compared with the easy part, the hard part is far more costly, which we need to calculate $f'^{\frac{\phi_k(p)}{r}}$.

On the construction of pairing-friendly curves, $p(x)$, $t(x)$, $r(x)$ are polynomials in $\mathbb{Q}[x]$, where $x\in\mathbb{Z}$ and t denotes the trace of π_p. One generally needs to choose a fixed integer x_0 such that both $p(x_0)$ and $r(x_0)$ are prime. Let $d=\dfrac{\phi_k(p)}{r}$. We usually represent the polynomial $d(x)$ in base $p(x)$. Then, f^d can be efficiently computed by some inexpensive cost of the Frobenius operations with a vectorial addition (-subtraction) chain [10, 5]. The shortest vectorial addition chain can guide implementers to compute the final exponentiation quickly. Another method to find the shortest vectorial addition chain is called as the latticed-based method [4]. This method computes $f^{d'}$ instead of f^d, where $d'=m\cdot d$ such that $r\nmid d'$, which makes the hard part be cheaper than the previous one. Hence, there exists a matrix M which contains some linear combinations of polynomials $d(x), xd(x), \cdots, x^{\deg(p)-1}d(x)$. The application of the LLL algorithm [8] for M can help implementers get an integer basis of M which has small entries. In addition, there is a method we use to compute the hard part of the final exponentiation via cyclotomic polynomials [12]. Recently, Hayashida et al. [6] defined homogeneous cyclotomic polynomials which avoid expanding $d(x)$ by $p(x)$ and achieve a good performance on the BLS families. Unfortunately, this method can not contribute to the final exponentiation on the KSS18 curve.

3 Faster Final Exponentiation for the KSS18 Curve

In this work, the embedding degree $k = 18$, then we only need to calculate $f^{(p^9-1)(p^3+1)} \to f$ in the easy part due to $\phi_{18}(p) = p^6 - p^3 + 1$ [7], which requires a multiplication, an inversion, a conjugation and three p^3−power Frobenius operations. The state-of-the-art method for computing the hard part of the final exponentiation on the KSS18 curve is proposed by Juan E. et al. [5], which requires 7 exponentiations by x, 35 Frobenius operations, 53 multiplications and 6 squarings. Our method is based on their work.

Note that $\varphi(18) = 6$ and $\deg(p) = 8$. After the application of the latticed-based method, we have

$$d'(x) = \sum_{i=0}^{5} \lambda_i(x) p(x)^i,$$

where $\lambda_i(x) = \sum_{j=0}^{7} \lambda_{ij} x^i$, $\lambda_{ij} \in \mathbb{Z}$. We use $\varphi(\cdot)$ to denote the Euler Totient function. The explicit formulae of $\lambda_i(x)$ are shown as follows.

$$\begin{aligned}
\lambda_0 &= 147x + 108x^2 + 21x^3 + 7x^4 + 5x^5 + x^6,\\
\lambda_1 &= -686 - 505x - 98x^2 - 35x^3 - 25x^4 - 5x^5,\\
\lambda_2 &= 6 - 133x^2 - 98x^3 - 19x^4 - 7x^5 - 5x^6 - x^7,\\
\lambda_3 &= 245x + 181x^2 + 35x^3 + 14x^4 + 10x^5 + 2x^6,\\
\lambda_4 &= -343 - 254x - 49x^2 - 21x^3 - 15x^4 - 3x^5,\\
\lambda_5 &= 3 + 7x^2 + 5x^3 + x^4.
\end{aligned}$$

Instead of finding the shortest vectorial addition-subtraction chain, we explore the relationship among the polynomials $\lambda_i(x)$.

Firstly, We start with the polynomial $\lambda_6(x) = x^2 + 5x + 7$. Then the other polynomials $\lambda_i(x)$ $(i = 0, 1, \cdots, 5)$ can be given below.

$$\begin{aligned}
\lambda_5 &= x^2\lambda_6 + 3,\\
\lambda_4 &= -3x \cdot \lambda_5 - 49\lambda_6,\\
\lambda_3 &= 2x^2 \cdot \lambda_5 + 35x \cdot \lambda_6,\\
\lambda_1 &= 2\lambda_4 + x \cdot \lambda_5,\\
\lambda_0 &= 2\lambda_3 + x \cdot \lambda_4,\\
\lambda_2 &= -x \cdot \lambda_0 + 2\lambda_5.
\end{aligned}$$

Hence, we require 37 factors to compute the final exponentiation, including

$$
\begin{aligned}
&f^2, f^3, f^5, f^7, f^x \longrightarrow f^{x+5} \longrightarrow f^{x^2+5x} \longrightarrow f^{x^2+5x+7} = f^{\lambda_6},\\
&f^{2\lambda_6} \longrightarrow f^{3\lambda_6} \longrightarrow f^{6\lambda_6} \longrightarrow f^{7\lambda_6}, f^{14\lambda_6},\\
&f^{x\lambda_6}, f^{x^2\lambda_6}, f^{x^2\lambda_6+3} = f^{\lambda_5}, f^{x\lambda_5}, f^{x^2\lambda_5},\\
&f^{2x\lambda_6} \longrightarrow f^{3x\lambda_6} \longrightarrow f^{6x\lambda_6} \longrightarrow f^{7x\lambda_6}, f^{14x\lambda_6}, f^{21x\lambda_6},\\
&f^{x\lambda_5+14\lambda_6} \longrightarrow f^{2(x\lambda_5+14\lambda_6)} \longrightarrow f^{3(x\lambda_5+14\lambda_6)},\\
&\overline{f}^{3x\lambda_5+49\lambda_6} = f^{\lambda_4}, f^{2\lambda_4}, f^{2\lambda_4+x\lambda_5} = f^{\lambda_1},\\
&f^{x^2\lambda_5+14x\lambda_6}, f^{2(x^2\lambda_5+14x\lambda_6)}, f^{x^2\lambda_5+35x\lambda_6} = f^{\lambda_3},\\
&f^{x^2\lambda_5+21x\lambda_6} = f^{\lambda_0}, \overline{f}^{x\lambda_0}, f^{2\lambda_5}, f^{-x\lambda_0+2\lambda_5} = f^{\lambda_2}.
\end{aligned}
$$

Here, We denote the conjugation of f by $\overline{f}$. Finally, the value of $f^{d'}$ can be computed by

$$f^{\lambda_0} \cdot (f^{\lambda_1})^p \cdot (f^{\lambda_2} \cdot (f^{\lambda_3})^p)^{p^2} \cdot (f^{\lambda_4} \cdot (f^{\lambda_5})^p)^{p^4}.$$

The detailed algorithm is shown in Algorithm 1.

Algorithm 1 Faster Final Exponentiation for KSS Curve with $k = 18$

```
INPUT: $f \in \mathbb{G}_{\Phi_{18}(p)}, x \in \mathbb{Z}$
OUTPUT: $f^{\frac{\Phi_k(p)}{r}} \in \mathbb{F}_{p^k}$
 1: $t_0 = f^x$;
 2: $t_1 = f^2$;
 3: $t_4 = f \cdot t_1$;
 4: $t_2 = t_1 \cdot t_4$;
 5: $t_1 = t_1 \cdot t_2$;
 6: $t_2 = (t_0 \cdot t_2)^x$;
 7: $c = t_1 \cdot t_2$;
 8: $t_0 = c^7$;
 9: $t_1 = t_0^2$;
10: $t_3 = \overline{c^x}$;
11: $c = \overline{t_3{}^x} \cdot t_4$;
12: $t_2 = \overline{c^x}$;
13: $t_4 = \overline{t_2}$;
14: $t_1 = (t_4 \cdot t_1)^3 \cdot t_0$;
15: $t_2 = \overline{t_2{}^x}$;
16: $t_0 = \overline{t_1}$;
17: $t_1 = (t_0{}^2 \cdot t_4)^p$;
18: $t_4 = t_1 \cdot (c^p \cdot t_0)^{p^4}$;
19: $t_3 = \overline{t_3}^7$;
20: $t_1 = t_3{}^2$;
21: $t_0 = (t_1 \cdot t_2)^2 \cdot t_3$;
22: $t_3 = t_1 \cdot t_3$;
23: $t_1 = t_2 \cdot t_3$;
24: $t_4 = t_1 \cdot t_4$;
25: $t_1 = \overline{t_1{}^x}$;
26: $t_2 = c^2$;
27: $t_1 = t_1 \cdot t_2$;
28: $t_0 = (t_0{}^p \cdot t_1)^{p^2}$;
29: $c = t_4 \cdot t_0$;
30: return $c$
```

4 Comparison and Conclusion

We draw comparisons between the original state-of-the-art method [5] and the new method for computing the final exponentiation on the KSS18 curve. Both methods are implemented by the C programming language which compiled with the GCC version 7.4.0. The code is tested within the RELIC library [1], and we use the Intel Core i7-8550U CPU processor at 1.80 GHz that runs on a 64-bit Linux.

The related parameters of the 676-bit KSS18 curve can be found in [3]. We neglect the inversions in $\mathbb{G}_{\phi_{18}(p)}$ since it only takes 9 negations. To get a steady result, the value of timings are the average of 10^4 runs. Here M, I and C are the cost of a multiplication, an inversion and a conjugation in $\mathbb{F}_{p^{18}}$, respectively. The cost of the p- power Frobenius operation and an exponentiation by x is denoted by F and E, respectively. Note that in the subgroup $\mathbb{G}_{\phi_{18}(p)}$, the squaring operation, which we denote by $\tilde{S}$, is cheaper than a squaring in $\mathbb{F}_{p^{18}}$. We remark from the results in Table 1 that at 192-bit security level,

Table 1. Operations and timings comparison for the final exponentiation on the KSS18 curve

Cost	Method	
	Method in [5]	This work
Operations	$I+M+9F+C$ $7E+53M+6\tilde{S}+35F$	$I+M+9F+C$ $7E+24M+11\tilde{S}+9F$
Clock Cycle $(\times 10^3)$	$12,982$	$11,849$
Time (ms)	6.52	5.95

our method is 8.74% faster than the previous record for the computation of the final exponentiation on the KSS18 curve.

Acknowledgments

The work of Chang-An Zhao is partially supported by the Major Program of Guangdong Basic and Applied Research under Grant No. 2019B030302008 and NSFC under Grant No. 61972428. The work of Zhi Hu is supported by the National Natural Science Foundation of China (Grant No. 61972420, No. 61602526) and Hunan Provincial Natural Science Foundation of China (2020JJ3050, 2019JJ50827).

References

1. Aranha, D.F., Gouvêa, C.P.L.: RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic
2. Aranha, D.F., Fuentes-Castañeda, L., Knapp, E., Menezes, A., Rodríguez-Henríquez, F.: Implementing pairings at the 192-bit security level. In: Abdalla, M., Lange, T. (eds.) Pairing-Based Cryptography – Pairing 2012. pp. 177–195. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
3. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. Journal of Cryptology **32**(1), 1–39 (2018)
4. Fuentes-Castañeda, L., Knapp, E., Rodríguez-Henríquez, F.: Faster hashing to $\mathbb{G}_2$. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography. pp. 412–430. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

5. Guzmán-Trampe, J.E., Cruz-Cortés, N., Dominguez Perez, L.J., Ortiz-Arroyo, D., Rodríguez-Henríquez, F.: Low-cost addition–subtraction sequences for the final exponentiation in pairings. Finite Fields and Their Applications **29**, 1–17 (2014)
6. Hayashida, D., Hayasaka, K., Teruya, T.: Efficient final exponentiation via cyclotomic structure for pairings over families of elliptic curves. Cryptology ePrint Archive, Report 2020/875 (2020), https://eprint.iacr.org/2020/875
7. Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N.P. (ed.) Cryptography and Coding. pp. 13–36. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
8. Lenstra, H.W. jr., L.A.L.L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**, 515–534 (1982)
9. Paterson, K.G.: Cryptography from Pairings - Advances in Elliptic Curve Cryptography. Cambridge University Press (2005)
10. Scott, M., Benger, N., Charlemagne, M., Dominguez Perez, L.J., Kachisa, E.J.: On the final exponentiation for calculating pairings on ordinary elliptic curves. In: Shacham, H., Waters, B. (eds.) Pairing-Based Cryptography – Pairing 2009. pp. 78–88. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
11. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory **56**(1), 455–461 (2010)
12. Zhang, X., Lin, D.: Analysis of optimum pairing products at high security levels. In: Galbraith, S., Nandi, M. (eds.) Progress in Cryptology - INDOCRYPT 2012. pp. 412–430. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
13. Zhao, C.A., Zhang, F.G., Huang, J.W.: All pairings are in a group. IEICE Transactions **91-A**(10), 3084–3087 (2008)