

Speeding Up Pairing Computation

Colm Ó hÉigearthaigh

School of Computing, Dublin City University.
Ballymun, Dublin 9, Ireland.
coheigearthaigh@computing.dcu.ie

Abstract. In this note, we describe how to achieve a simple yet substantial speed up of Miller's algorithm, when not using denominator elimination, and working over quadratic extension fields.

Keywords: Tate pairing, pairing-based cryptosystem

1 Introduction

Following a seminal paper by Boneh and Franklin [2] in 2001, there has been an explosion of interest in the exploitation of the properties of bilinear pairings on elliptic curves for cryptographic protocols. Naturally, there has also been much focus on the efficient implementation of pairings. Victor Miller gave the first algorithm [5] for computing a bilinear pairing, specifically the *Weil* pairing. However in practice the *Tate* pairing is used, as it is computationally less expensive.

In an important paper, Barreto et. al. [1] gave criteria under which divisions in Miller's algorithm can be eliminated entirely. According to [6], this reduces the calculation time by almost 50%.

In this paper, we show how to speed up the algorithm when we are working over quadratic extension fields (as we nearly always are for efficiency reasons), when not using the denominator elimination idea of Barreto et. al. Although denominator elimination is still slightly faster, we show that the difference is not as large as is generally thought.

2 Miller's Algorithm

Miller's algorithm is described in Algorithm 1 using two key optimisations introduced by Galbraith et. al. in [4]. Firstly, the first point P is defined over the base field \mathbb{F}_p rather than over the larger field \mathbb{F}_{p^k} , where k is the embedding degree of the curve in question. Secondly, to avoid performing a division each loop iteration, the divisions are postponed until the end of the loop. To do this we use two variables in the loop, to effectively replace a division with a squaring each loop iteration, which is considerably less expensive to compute.

After the main loop one performs the final exponentiation of $(p^k - 1)/r$ to obtain a unique value over \mathbb{F}_{p^k} , where r is the prime-order subgroup we are working with. It is

well known that if one implements arithmetic in \mathbb{F}_{p^k} using quadratic extensions, one can exponentiate an element in this field to the power of $p^{k/2}$ using a simple conjugation. Conjugation is denoted by $\bar{x} = (a - bi)$ for an element $x = (a + bi) \in \mathbb{F}_{p^k}$.

Taking advantage of this, it is standard to efficiently compute the final exponentiation as $f = \bar{f}/f$ followed by $f = f^{(p^{k/2}+1)/r}$, the latter of which can be computed efficiently using Lucas Sequences [7].

Algorithm 1 Miller's algorithm

INPUT: $P \in E(\mathbb{F}_p), Q \in E(\mathbb{F}_{p^k})$ where P has order r .

OUTPUT: $\langle P, Q \rangle_r$

```

1:  $f_c \leftarrow 1, f_d \leftarrow 1$ 
2:  $T \leftarrow P$ 
3: for  $i \leftarrow \lfloor \log_2(r) \rfloor - 1$  downto 0 do
4:    $\triangleright$  Calculate lines  $l$  and  $v$  in doubling  $T$ 
5:    $T \leftarrow [2]T$ 
6:    $f_c \leftarrow f_c^2 \cdot l(Q)$ 
7:    $f_d \leftarrow f_d^2 \cdot v(Q)$ 
8:   if  $r_i = 1$  then
9:      $\triangleright$  Calculate lines  $l$  and  $v$  in adding  $P$  to  $T$ 
10:     $T \leftarrow T + P$ 
11:     $f_c \leftarrow f_c \cdot l(Q)$ 
12:     $f_d \leftarrow f_d \cdot v(Q)$ 
13:   end if
14: end for
15:  $f \leftarrow f_c/f_d$ 
16:  $f \leftarrow \bar{f}/f$ 
17:  $f \leftarrow f^{(p^{k/2}+1)/r}$ 
18: return  $f$ 

```

3 Our Idea

In this section, we present a way to optimise the final exponentiation for arbitrary pairing calculation over quadratic extension fields. This is analogous to a technique mentioned in [3] (section 16.5.2), however we go beyond this and show how we can use our idea to reduce the arithmetic in the main loop of Miller's algorithm in a simple way, when one is not using denominator elimination. We also save a multiplication over \mathbb{F}_{p^k} after the loop.

Looking at steps 15 and 16 after the main loop in Algorithm 1, one can combine them to get; $f = (\bar{f}_c f_d)/(f_c \bar{f}_d)$. Then one can eliminate a multiplication by noticing that the denominator is just the conjugate of the numerator, ie. $f_c \bar{f}_d = \overline{\bar{f}_c f_d}$. Therefore, we can write the computation after the main loop as;

$$f = f_c \bar{f}_d$$

$$f = \bar{f}/f$$

followed by the final $f^{(p^{k/2}+1)/r}$ exponentiation. So, as the conjugation operation is effectively free to compute, we have saved a multiplication.

As is mentioned in the previous section, one uses two variables f_c and f_d to avoid divisions in the main loop of Miller's algorithm. However we can use the formulae we derived for the final exponentiation to avoid divisions in the main loop and also to avoid using two variables. As we have replaced the $f = f_c/f_d$ operation with $f = f_c\overline{f_d}$, effectively eliminating the division by absorbing it into the powering, we can push the evaluation of $f = f_c\overline{f_d}$ back into the main loop in the algorithm.

So, in the main loop we can have just one variable f , and we multiply it by the function f_c and then by the conjugate of the function f_d . By doing this, we obviously also eliminate the $f = f_c\overline{f_d}$ multiplication at the end. The new algorithm is detailed in Algorithm 2. Note that simply removing the $v(Q)$ function from the algorithm gives the exact same algorithm as when one is using denominator elimination.

To see how our idea works using a more formal approach, we prove in the following lemma that inverting an element in a quadratic extension field is the same as conjugating it, so long one raises to the power of $(p^{k/2} - 1)$ afterwards, which is exactly what one does with the final exponentiation after the main loop in Miller's algorithm.

Lemma 1. *Let $x = (a + ib) \in \mathbb{F}_{p^k}$, where $a, b \in \mathbb{F}_{p^{k/2}}$. Then the following holds;*

$$\left(\frac{1}{x}\right)^{p^{k/2}-1} = (\overline{x})^{p^{k/2}-1}$$

Proof. $\left(\frac{1}{x}\right)^{p^{k/2}-1} = \frac{1}{(a+ib)^{p^{k/2}-1}} = \frac{(a+ib)}{(a-ib)}$. Similarly, $\overline{x}^{p^{k/2}-1} = \overline{(a+ib)^{p^{k/2}-1}} = (a-ib)^{p^{k/2}-1} = \frac{(a+ib)}{(a-ib)}$.

Therefore, as we have eliminated the variable f_d from the pairing calculation, we save a squaring over \mathbb{F}_{p^k} each iteration of the loop. This is still not as good as performing denominator elimination, which would save a multiplication over this again each iteration, yet it is still a noticeable implementational improvement when one does not use denominator elimination. It also shows that denominator elimination does not give quite as big an improvement as is generally thought.

4 Conclusion

We have improved the running time of Miller's algorithm over quadratic extension fields by removing a squaring from the main loop, and optimising the final exponentiation slightly. These tricks show that using denominator elimination is not quite as efficient as thought, the speedup gained is likely to be only around 30% for elliptic curves compared to the method in this paper rather than almost 50% reported in [6].

Finally, we note that our method may be more efficient than using denominator elimination for pairing computation on higher genus hyperelliptic curves. This is because with our method we can choose a divisor consisting of a single point instead of the more general divisor with g points on it, where g is the genus of the curve. Using denominator elimination may force us into using more general divisors, thus increasing the number of multiplications in the loop.

Algorithm 2 An improved Miller's algorithm

INPUT: $P \in E(\mathbb{F}_p), Q \in E(\mathbb{F}_{p^k})$ where P has order r .OUTPUT: $\langle P, Q \rangle_r$

```
1:  $f \leftarrow 1$ 
2:  $T \leftarrow P$ 
3: for  $i \leftarrow \lfloor \log_2(r) \rfloor - 1$  downto 0 do
4:    $\triangleright$  Calculate lines  $l$  and  $v$  in doubling  $T$ 
5:    $T \leftarrow [2]T$ 
6:    $f \leftarrow f^2 \cdot l(Q) \cdot \overline{v(Q)}$ 
7:   if  $r_i = 1$  then
8:      $\triangleright$  Calculate lines  $l$  and  $v$  in adding  $P$  to  $T$ 
9:      $T \leftarrow T + P$ 
10:     $f \leftarrow f \cdot l(Q) \cdot \overline{v(Q)}$ 
11:   end if
12: end for
13:  $f \leftarrow \overline{f}/f$ 
14:  $f \leftarrow f^{(p^{k/2}+1)/r}$ 
15: return  $f$ 
```

5 Acknowledgements

We would like to thank Mike Scott, Steven Galbraith, Caroline Sheedy and Noel McCullagh for comments on this paper.

References

1. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology – Crypto'2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer-Verlag, 2002.
2. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.
3. H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman & Hall/CRC, 2006.
4. S. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithmic Number Theory – ANTS V*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer-Verlag, 2002.
5. V. S. Miller. Short programs for functions on curves. Unpublished manuscript, 1986. <http://crypto.stanford.edu/miller/miller.pdf>.
6. M. Scott. Faster identity based encryption. *Electronics Letters*, 40(14):861, 2004.
7. M. Scott and P. Barreto. Compressed pairings. In *Advances in Cryptology – Crypto'2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 140–156. Springer-Verlag, 2004.